VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2017 Bc. Pavol Vrablic

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

SIMULACE SDN SÍTĚ
SIMULATION OF SDN NETWORK

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Pavol Vrablic**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Bohumil Novotný**
SUPERVISOR

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Telekomunikační a informační technika**
Ústav telekomunikací

Student: Bc. Pavol Vrablic ***ID:*** 154668
Ročník: 2 ***Akademický rok:*** 2016/17

NÁZEV TÉMATU:

Simulace SDN sítě

POKYNY PRO VYPRACOVÁNÍ:

V rámci diplomové práce prozkoumejte a analyzujte technologii SDN a navrhněte metody testování jejích parametrů. Pro simulace využijte operačního systému Allinone SDN App Development Starter VM. Pro dosažení a prezentaci výsledků využijte nástrojů obsažených v zadaném operačním systému.

DOPORUČENÁ LITERATURA:

[1] NADEAU, Thomas D. a Kenneth GRAY. SDN: software defined networks. Sebastopol, CA: O'Reilly Media, 2013. ISBN 978-1-4493-4230-2.

[2] GORANSSON, Paul a Chuck BLACK. Software defined networks: a comprehensive approach. Amsterdam: Morgan Kaufmann, c2014. ISBN 978-0-12-416675-2.

Termín zadání: 1.2.2017 ***Termín odevzdání:*** 24.5.2017

Vedoucí práce: Ing. Bohumil Novotný
Konzultant:

doc. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

Hlavným cieľom tejto práce je oboznámiť sa s technológiou softvérovo definovaných sietí a naučiť sa používať niektoré nástroje na meranie a simulovanie týchto sietí.

KĽÚČOVÉ SLOVÁ

SDN, softvérovo definovaná sieť, OpenFlow, OpenVSwitch, SDN kontrolér, virtualizácia, sieťová konvergencia, latencia, priepustnosť, Cbench, MT-Cbench

ABSTRACT

The main aim of this work is to become familiar with the technology of software-defined networks and learn to use some of the tools to measure and simulate these networks.

KEYWORDS

SDN, Software Defined Network, OpenFlow, OpenVSwitch, SDN controller, Virtualization, Network Convergence, Latency, Throughput, Cbench, MT-Cbench

VRABLIC, Pavol *Simulace SDN sítě*: diplomová práca. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2017. 66 s. Vedúci práce bol Ing. Bohumil Novotný.

VYHLÁSENIE

Vyhlasujem, že som svoju diplomovoú prácu na tému „Simulace SDN sítě“ vypracoval samostatne pod vedením vedúceho diplomovej práce, využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej diplomovej práce ďalej vyhlasujem, že v súvislosti s vytvorením tejto diplomovej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/alebo majetkových a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona Českej republiky č. 121/2000 Sb., o práve autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon), v znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhej, hlavy VI. diel 4 Trestného zákoníka Českej republiky č. 40/2009 Sb.

Brno ....

podpis autora

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic

http://www.six.feec.vutbr.cz

POĎAKOVANIE

Výzkum popísaný v tejto diplomovej práci bol realizovaný v laboratóriách podporených projektom SIX; registračné číslo CZ.1.05/2.1.00/03.0072, operačný program Výzkum a vývoj pro inovace.

Brno ....

podpis autora

OBSAH

ZOZNAM OBRÁZKOV

ZOZNAM TABULIEK

ÚVOD

V súčasnej dobe nástup cloudových služieb a virtualizácie spolu s väčšou potrebou mobility koncových zariadení zmenil nároky na sieť a jej fungovanie. Bolo treba vytvoriť niečo, čo by sa dokázalo prispôsobiť týmto novým potrebám a zároveň by dokázalo pracovať s obvyklou architektúrou siete, niečo čo by nestálo nemalé finančné prostriedky na prebudovanie. Tieto potreby dali možnosť vzniku softvérovo-definovaným sieťam, ktoré sa dokonale prispôsobili práve mobilite zariadení v cloude a zároveň implementáciou SDN protokolu do existujúcich zariadení aktualizáciou firmvéru stali ekonomicky prístupným riešením. Táto práca ukáže ako SDN sieť vlastne vyzerá a ako funguje s protokolom OpenFlow, ktorý je v tejto oblasti momentálne najviac rozšírený medzi voľne dostupnými protokolmi. Nasimulovaním siete ukáže funkčnosť a možnosti ktoré ponúka, a poukáže na parametre ktoré poskytuje a výkon ktorým disponuje SDN kontrolér, ktorý je jadrom tejto siete.

1 SOFTVÉROVO DEFINOVANÉ SIETE – SDN

Základným princípom SDN je nesporne oddelenie riadiacich a dátových rovín. Tento koncept nie je nový a súčasný vývoj často nahliada do pôvodných myšlienok. Už vtedy sa uvažovalo nad tým, ako ďaleko môže byť riadiaca rovina umiestnená od dátovej roviny, koľko inštancií je potrebných na udržanie odolnosti a vysokej dostupnosti takejto siete, ako riešiť presun riadiacej roviny a či je potreba logickej centralizácie riadiacej roviny. Výsledkom skúmania bolo striktne centralizovanie riadiacej roviny, čo prináša mnohé výhody pre sieťových operátorov.[2] Pre hlbšie pochopenie fungovania SDN, treba poznať architektúru typickej SDN siete, ktorá je opísaná v nasledujúcej kapitole.

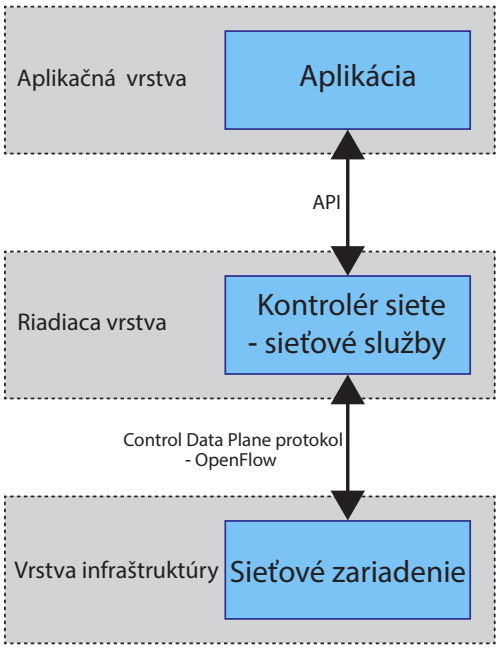
1.1 Architektúra siete

Ako už bolo spomenuté, podstatou SDN je oddelenie riadiacej roviny a dátovej roviny (Control Plane a Data Plane) od seba do dedikovaných zariadení.
Tieto roviny môžeme popísať takto:

- **Riadiaca rovina:** Logická časť siete, ktorá rozhoduje o riadení toku dát. Zaisťuje výmenu informácií o topológii s inými zariadeniami, vytvára smerovanie, jej funkciou je správa, výmena informácií, konfigurácia a signalizácia v sieti. V SDN sa stará o vytvorenie súboru pravidiel o preposielani paketov a tieto pravidlá vnucuje dátovej rovine. Riadiaca rovina ďalej udržuje sieťovú topológiu zvanú RIB (routing information base), ktorú udržuje konzistentnú a bez sieťových slučiek. FIB (forwarding information base), čo sú vlastne preposielacie tabuľky, sú zrkadlené medzi dátovou a riadiacou rovinou. FIB sa programuje automaticky potom, čo je zabezpečená konzistentnosť RIB. K vykonávaniu tejto činnosti je určený riadiaci program, ktorý obsahuje kontrolér.[2]
- **Dátová rovina:** Je časťou siete, ktorá predáva pakety z jedného portu na iný, a pritom sa riadi práve pravidlami riadiacej roviny. Pakety sú prijaté prostredníctvom série komunikácií a následne sú vykonané základné kontroly. Po dobre vytvorenom pakete sa tento spracováva v dátovej rovine nahliadnutím do FIB tabuľky, ktoré sú naprogramované riadiacou rovinou. Prichádza k veľmi rýchlemu spracovaniu paketov, pretože stačí identifikovať cieľ paketu vo FIB. Výnimkou spracovania je stav kedy sa zachytí paket s neznámym cieľom a ten je zaslaný na spracovanie do RIB. FIB tabuľky spracováva napríklad ASIC čip, ktorý sa stará o veľmi rýchle spracovávanie inštrukcií na úrovni hardvéru.[2]

Riadiaca rovina je teda programovateľná a je presunutá do samostatného zariadenia nazývaného kontrolér. Tým je nejaký počítač s príslušným výkonom k veľkosti, zložitosti a účelu siete. Dátová rovina zostáva naďalej v prepínačoch. Zariadenia rôznych výrobcov sú vďaka tomu riadené jednotným systémom a sieť zložená v infraštruktúre z proprietárnych systémov sa javí ako jednotná.

Architektúra SDN sa delí na 3 vrstvy, kde každá vrstva komunikuje len so svojou susednou vrstvou. Týmito vrstvami je vrstva aplikačná, pod ňou vrstva riadiaca a ako spodná vrstva je vrstva infraštruktúry.



Obr. 1.1: Architektúra SDN siete

1.1.1 Aplikačná vrstva

Najvyššia vrstva celej architektúry, v ktorej sa nachádza celá inteligencia SDN. Zahrňuje vrstvy 4. až 7. ISO/OSI 7-vrstvového modelu. Túto vrstvu tvoria aplikácie naprogramované vyšším programovacím jazykom, najčastejšie je to JAVA, C++ či Python. Tieto aplikácie priamo komunikujú so sieťovými prvkami a ovplyvňujú ich činnosť a konfiguráciu. Využíva sa abstraktný pohľad na sieť, ktorý sprostredkuje kontrolér. Aplikácie pomocou API cez takzvaný „southbound interface“ následne:

- zaisťujú kvalitu služieb,
- riešia rozloženie záťaže,
- monitorujú sieťovú prevádzku,
- konfigurujú prvky siete alebo kontrolér,
- stanovujú pravidlá prístupu ACL (Access Control List),
- riešia výpadky siete alebo slučky v sieti.

1.1.2 Riadiaca vrstva

Prostredná vrstva architektúry SDN, ktorá sprostredkuje komunikáciu medzi aplikáciami v aplikačnej vrstve a hardvérom vo vrstve infraštruktúry je riadiaca vrstva. Hlavným prv-

kom riadiacej vrstvy je kontrolér alebo viac kontrolérov. Software tvoriaci kontrolér pozná topológiu siete a považuje prvky siete z abstraktného pohľadu za rovnaké prvky riadiace sa rovnakou logikou. Kontrolér je v podstate len softvér, ktorý môže bežať na akomkoľvek počítači s vhodnou konfiguráciou hardvéru. Často sú kontroléry z pohľadu hardvéru konštruované ako dedikované zariadenia, ktoré sú optimalizované čisto na účel riadenia SDN. Sú to väčšinou servery na báze Linux a očakáva sa u nich aspoň priemerný výkon. [12]

Kontrolér prebral kontrolu riadiacej roviny zo zariadení vo vrstve infraštruktúry. Rozhoduje o nakladaní s dátovými tokmi pomocou aplikácií vo vyššej vrstve. Pracuje hneď v niekoľkých rovinách. Požiadavky z prepínačov sú spracovávané cez takzvané „southbound“ rozhranie a aplikácie komunikujú s jadrom kontroléru cez rozhranie zvané „northbound“. Kontrolér má v SDN veľmi dôležitú funkciu a často sa v sieti preto využívajú redundantné kontroléry. Kontrolér je totiž citlivým miestom v sieti. V prípade výpadku či útoku naň by totiž mohlo prísť k znefunkčneniu celej siete. Na komunikáciu medzi kontrolérmi existuje rozhranie zvané „east-west“. [12]

1.1.3 Vrstva infraštruktúry

Túto vrstvu tvoria u SDN sieti prepínače a smerovače. Väčšina sieti v súčasnosti prenáša užívateľské dáta ale aj rôzne protokoly ktoré slúžia k správe a riadeniu siete. Typicky sú to napríklad protokoly BGP, OSPF, RIP, RSTP, ICMP. Implementácia týchto protokolov býva často mierne odlišná u každého výrobcu zariadení. Často sa potom stáva že rozšíriteľnosť a škálovateľnosť siete naráža práve na limity kompatibility. Sieť sa stáva náročnejšou na správu a konfiguráciu. U klasických sieti bola riadiaca a dátová rovina obsiahnutá len v týchto zariadeniach. U SDN sieti prenesením riadiacej roviny na kontrolér sme znížili nároky na inteligenciu zariadenia. Tieto zariadenia, takzvané hlúpe prepínače, kladú menšie nároky na výrobu — jednoduchší software. V súčasnosti sa ale SDN protokoly integrujú aj do existujúcich zariadení aktualizáciami firmvéru vďaka čomu môžeme postupne prebudovať aj sieť so súčasnou infraštruktúrou. [12]

Zariadení použiteľných vo vrstve infraštruktúry s funkčnou podporou pre SDN siete je v predaji hneď niekoľko:

- A10 Networks – AX Series
- Brocade CER 2000, CES 2000, MLX Series
- Extreme Networks – BlackDiamond 8000, X8 Series, Summit X670
- HP 2920, 3500, 3800, 5400 Switch Series
- IBM RackSwitch™ G8264T
- NEC Hybrid OpenFlow Switch PF5240, OpenFlow® Switch PF5820
- Pica8 Open Switches,
- Plexxi Switch 1

1.2 Vlastnosti SDN sieti

1.2.1 Módy fungovania

SDN siete majú tri módy fungovania, v ktorých rôznymi spôsobmi prichádza k narábaniu s dátovými jednotkami:

- **Proaktívny mód:** Je režim, pri ktorom kontrolér nenastaví prepínač v reakcií na paket, ale distribuuje do prepínačov vopred pripravené pravidlá, ktorými sa bude prepínač riadiť. Využíva sa toho napríklad vo virtualizačnom prostredí pri nastavovaní virtuálneho serveru a jeho siete. Prvotné toky hneď spracuje TCAM, čo sa deje veľmi rýchlo. Proaktívnym režimom sa je teda možné vyhnúť latencii, ktorú do prenosu vnáša konzultácia prepínača s kontrolérom o každom novom dátovom toku. [9]
- **Reaktívny mód:** Toto je režim, kde prepínač notifikuje kontrolér o každom novom toku, s ktorým nevie ako narábať, v riadiacej správe zvanej `Packet_In` a `Packet_Out` v porovnaní s proaktívnym režimom, kedy majú pakety vždy v tabuľke tokov príslušné pravidlá. Pri veľkom počte zariadení v sieti táto konfigurácia značne zaťažuje kontrolér a môže ho dokonca preťažiť, čo vedie k stratovosti a nespoľahlivosti. Preto treba túto možnosť využiť len v limitovaných podmienkach menších sieti, kde sú rádovo desiatky prepínačov alebo len v prípade kedy potrebujeme mať úplnú kontrolu nad dátovou prevádzkou siete. Zo správ `Packet_In` je možné navyše vypočítať latenciu cesty a netreba využívať klasické protokoly, počítajúc cenu cesty ktoré majú dlhšiu dobu konvergencie. [9]
- **Hybridný mód:** Je kombináciou proaktívneho a reaktívneho módu. Je vhodný pri aplikáciach, kde potrebujeme udržať flexibilitu riadenia toku dát v sieti a zároveň dosahovať rýchle zasielania paketov pri nízkom oneskorení. V podnikovom prostredí sa vyžaduje často mať sieť s nízkou latenciou, ktorú môžeme skombinovať s bezpečnosťou, ktorú nám SDN poskytuje. Okrem kombinácie týchto dvoch metód môžeme navyše využívať aj lokálnu riadiacu rovinu v prepínačoch. Vďaka tomu môžeme využívať stávajúcu sieť, kde prvky fungujú klasickým spôsobom ale do siete môžeme pridať rôzne SDN funkcie. [9]

1.2.2 Výhody využívania SDN

Väčšina technológií ktorá vznikne, má svoje výhody a nevýhody. Úspešnosť vývoja SDN zvyšuje počet výhod, ktorý prevyšuje počet nevýhod.
Medzi podstatné výhody SDN treba zaradiť nasledovné vlastnosti:

- **Unifikácia zariadení:** V celej sieti sa zariadenia rôznych výrobcov správajú rovnako a z abstraktného pohľadu na sieť je to sieť prepínačov OpenFlow. Toto však platí iba ak zariadenia využívajú a podporujú rovnakú verziu protokolu, ktorý prechádza stále vývojom.
- **Zjednodušenie zložitosti:** Naprogramovanie nejakej aplikácie ktorá zautomatizuje činnosť, ktorú je treba bežne nastaviť manuálne v závislosti napríklad na vyťažení

liniek.

- **Centralizované ovládanie:** Celú sieť je možné spravovať z centrálneho miesta. Pri re-konfigurácií sa netreba prihlasovať do každého prvku siete samostatne a orientovať sa v rozdielnych manažmentoch konfigurácie.
- **Bezpečnosť:** Vďaka centrálnej správe máme celkový prehľad nad konfiguráciou celej siete. Navyše nasadením rôznych aplikácií môžeme kontrolovať podozrivé správanie na sieti.
- **Spoľahlivosť:** Pri presune užívateľskej služby netreba prekonfigurovať celú sieť. SDN sieť môžeme nakonfigurovať tak že reaguje pružne na takéto zmeny.
- **Zaistenie kvality služieb:** Aplikácia kontroléru môže monitorovať sieť a dynamicky ju prekonfigurovať na základe vyťaženia a odozvy. [11]

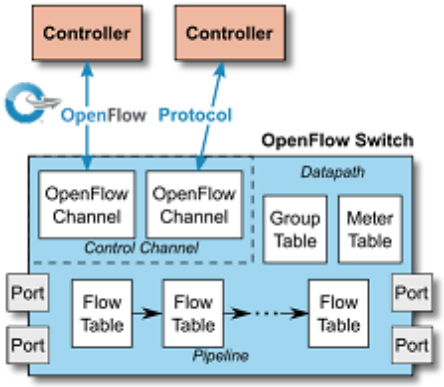
1.2.3 Nevýhody využívania SDN

Nevýhody, ktoré môže zavedenie SDN priniesť, sú závislé od prostredia do ktorého sú nasadené. Ak opomenieme nasadenie SDN do veľmi malých sieti, kde celú topológiu tvorí zopár zariadení a neočakávame využívanie nadštandardných funkcií, pri tých väčších musíme myslieť na tieto veci:

- **Neštandardizovaný kontrolér:** Ani jeden kontrolér sa zatiaľ nestal nejakým štandardom, z ktorého by mohli vychádzať vývojári pri vývoji svojich SDN aplikácií. Následkom tohto je, že aplikácie nie sú prenositeľné a univerzálne medzi rôznymi kontrolérmi. Tento problém sa snaží riešiť organizácia *Opendaylight-project*, ktorá sa práve takýto kontrolér snaží vyvinúť.[2]
- **Vývoj:** Organizácia *Open Networking Foundation*, ktorá vytvára špecifikácie protokolu OpenFlow sa skladá momentálne z približne 150 členov, medzi ktorými sú aj poskytovatelia služieb ako Google, Facebook či Microsoft. Tí však pri nasadzovaní vlastných SDN sieti prichádzajú s vlastnými modifikáciami protokolu OpenFlow a hardvérom optimalizovaným na jeho použitie. Tieto riešenia sú neverejné. Vznikajú tak rôzne riešenia ktoré sa môžu v malej či väčšej miere líšiť od zavedeného štandardu a zároveň sa tak môže spomaliť vývoj tohto protokolu.
- **Zložitosť:** Vytvoriť topológiu trvá dlhšie než ako sme boli zvyknutý predtým. Je to hlavne preto, že technológia je stále vo vývoji a nemá nejaké ucelené prostredie, v ktorom je možné všetko nastaviť, navyše často sa toto prostredie mení. Navyše je treba poznať nejaký programovací jazyk (Python, Java, C). Odporuje to tak trochu jednej z požiadavok na sieť – jednoduchosť správy.

2 ZÁKLADY PROTOKOLU OPENFLOW

V dobe písania tejto práce je verzia 1.5.1 poslednou špecifikáciou, ktorú spracovala organizácia ONF. Táto špecifikácia zahŕňa požiadavky na komponenty a základné funkcie prepínača a zároveň protokol, ktorým kontrolér tento prepínač riadi.



Obr. 2.1: Komponenty OpenFlow [1]

OpenFlow switch je logický prepínač, ktorý obsahuje jednu či viac tabuliek tokov (flow table), tabuľku skupín (group table), tabuľku merania (meter table) a jeden alebo viac kanálov ku kontroléru.

Použitím protokolu OpenFlow môže kontrolér pridávať, aktualizovať a mazať záznamy tokov v tabuľkách tokov, a to reaktívne v reakcií na paket, alebo proaktívne. Každá tabuľka tokov obsahuje niekoľko záznamov tokov, každý záznam pozostáva z niekoľkých porovnávacích polí, čítačov a inštrukcií určených na aplikovanie na pakety vyhovujúcich zhode.

Porovnávanie začína prvou tabuľkou a môže pokračovať na ďalšie zreťazené tabuľky. Paket sa porovnáva v poradí podľa priority záznamu v každej tabuľke a použije sa prvý nájdený záznam. Ak je nájdený, vykonajú sa inštrukcie asociované s týmto záznamom. Ak nie je nájdený, ďalšie spracovanie sa riadi tabuľkou *table-miss*. Paket sa vtedy pošle kontroléru cez OpenFlow kanál, zahodí sa, alebo pokračuje do ďalšej porovnávacej tabuľky tokov.

Inštrukcia asociovaná s každým záznamom toku v tabuľke obsahuje akciu, v ktorej je popísané ako sa má paket preposlať, modifikovať alebo spracovať skupinovou tabuľkou. Inštrukcie umožňujú spracovanie v tabuľkách po sebe, pričom sa môžu medzi tabuľkami využívať na komunikáciu metadáta z informácií z predchádzajúcich tabuliek. Spracovanie paketu sa končí v momente, kedy inštrukcia v zázname toku neobsahuje ďalšiu tabuľku.

Záznamy tokov môžu smerovať na rôzne porty. Zvyčajne je to fyzický port ale môže to byť aj logický port definovaný prepínačom, alebo aj rezervovaný port definovaný špecifikáciou. Logickým portom môže byť napríklad skupina agregovaných liniek, tunel, či

loopback. Rezervovaný port môže vykonať akcie typu: poslanie paketu do kontroléra, záplavové šírenie paketu či zaslanie pakety na obyčajný port, ktorý nespadá pod OpenFlow. Tu ho prepínač spracuje obvyklým procesom.

Akcie asociované so záznamami tokov môžu nasmerovať paket priamo do skupinovej tabuľky, ktorá definuje ďalšie spracovanie. Tohto sa využíva pri komplexnejšom preposielani paketov, ako napríklad pri agregácií liniek a pri dynamickom smerovaní liniek. Skupina navyše môže všetky pakety smerovať na jeden identifikátor.

Skupinová tabuľka obsahuje záznamy skupín, každý záznam obsahuje zoznam skupín akcií k vykonaniu podľa typu použitej skupiny. Akcie ktoré obsahujú skupiny akcií sú aplikované na všetky pakety zaslané do danej skupiny.[1]

2.1 Typy správ

Zariadenia v sieti komunikujú protokolom OpenFlow cez kontrolný kanál s kontrolérom prostredníctvom rôznych druhov správ. Podľa smeru komunikácie sa delia na tri základné skupiny, symetrické, asynchrónne a správy kontrolér-prepínač.

2.1.1 Symetrické správy

Symetrické správy sú správy, ktoré si môže kontrolér a prepínač posielať medzi sebou navzájom. Často preto bývajú označované aj ako servisné správy.

Typy správ ktoré sem patria:

- **Hello:** Tieto správy sa posielajú medzi prepínačom a kontrolérom pri zostavovaní spojenia.
- **Echo:** Echo žiadosti a odpovede sú posielané či už z prepínača alebo kontroléru a musia vracať odpovede. Využívajú sa na kontrolu funkčnosti spojenia a na meranie šírky pásma a oneskorenia siete.
- **Error:** Aj tieto správy môžu zasielať obe zariadenia. Slúžia k notifikácií o svojich problémoch druhej strane spojenia. Prepínač ich najčastejšie posiela ak príde k poruche vykonania žiadosti, ktorú inicializoval kontrolér.
- **Experimenter:** Sú to správy, ktoré nemajú momentálne iný účel ako testovanie. Používajú sa pri vývoji nových funkcií protokolu OpenFlow.[1]

2.1.2 Asynchrónne správy

Asynchrónne správy sú správy, ktoré posiela prepínač kontroléru. Sú to dotazy na pravidlá zachádzania s dátovými jednotkami, informácie o stavoch portov a informácie o vyťažení.

Hlavné typy asynchrónnych správ ktoré sa používajú:

- **Packet-In:** Pomocou tejto správy sa prenáša kontrola paketu z prepínača do kontroléra. Správa sa generuje keď prepínač obdrží paket, ku ktorému nevlastní zodpovedajúcu tabuľku tokov, resp. zodpovedajúci záznam toku a vtedy, keď jej prepínač nastavený do režimu, kedy všetky pakety zasiela na kontrolér, s tým že kontrolér má rozhodnúť o tom čo sa s nimi bude diať. Kontrolér naspäť odpovedá správami *Packet-Out*. V tých sú informácie ako má prepínač s paketmi naložiť.
- **Flow-Removed:** Informuje kontrolér o odstránení záznamu toku z tabuľky tokov. Toto sa deje len pri záznamoch ktoré majú príznak `OFPFF_SEND_FLOW_REM`. Správy sú generované potom, čo kontrolér požiada o zmazanie nejakého záznamu, po expirovaní záznamu ktorému vypršal *timeout* a iné...
- **Port-Status:** Informuje kontrolér o zmenách na porte. Očakáva sa od prepínača zasielanie tejto správy všetkým kontrolérom, ak sa zmení konfigurácia portu alebo jeho stav. Zahrnuté sú v nej zmeny na porte inicializované užívateľom manuálne ale aj zmeny spôsobené linkou (*Up/Down*).
- **Role-Status:** Informuje kontrolér o zmene svojej role. Keď nový kontrolér zvolí seba za *MASTER*, očakáva sa od prepínača táto správa, ktorá je zaslaná pôvodnému *masterovi*.[1]
- **Controller-Status:** Informuje kontrolér o zmenách na OpenFlow kanáli. Prepínač informuje všetky kontroléry, ak sa zmení akýkoľvek kanál k niektorému z prepínačov. Táto správa pomáha vyriešiť takzvaný „failover“, keď stratia kontroléry schopnosť navzájom medzi sebou komunikovať.
- **Flow-monitor:** Informuje kontrolér o zmene v tabuľke tokov. Kontrolér môže definovať niekoľko monitorov pre sledovanie zmien v tabuľkách tokov.[1]

2.1.3 Správy typu kontrolér-prepínač

Ako je možné spoznať podľa názvu, jedná sa o správy ktoré sú odosielané výhradne kontrolérom smerom k prepínaču. Kontrolér odosiela dotaz a môže očakávať ale aj nemusí odpoveď. Vďaka odpovediam dokáže kontrolér sledovať stav zariadení, ktoré má vo svojej správe a podľa toho prispôsobovať ich konfiguráciu.

Protokol OpenFlow momentálne pozná tieto správy typu kontrolér-prepínač:

- **Features:** Táto žiadosť vyžiada identifikáciu prepínača a informácie o jeho schopnostiach. Prepínač odpovedá správou **Features reply**, v ktorej sú tieto informácie obsiahnuté. Tieto správy sa zasielajú pri prvotnom nadväzovaní spojenia.
- **Configuration:** Týmto parametrom sa kontrolér dotazuje prepínača na nastavenia alebo nimi nastavuje.
- **Modify-State:** Týmito správami sa nastavujú, modifikujú či mažú pravidlá v prietokových tabuľkách, prípadne sa modifikujú stavy portov.
- **Read-State:** Touto správou je možné čítať rôzne stavy v prepínačoch ako štatistiky, konfigurácia, schopnosti.

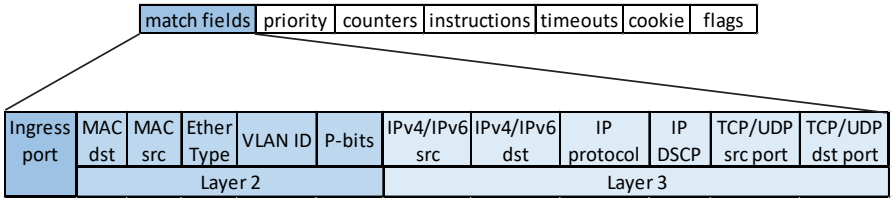
- **Packet-Out:** Táto správa je odpoveďou na správu `Packet-In`. Obsahuje celý paket alebo identifikátor paketu podľa ktorého vie prepínač určiť o ktorý paket sa jedná. Okrem toho obsahuje postupnosť akcií, ktoré sa majú s paketom vykonať. Ak je prázdny, je to pokyn pre prepínač, že má byť paket zahodený.
- **Barrier:** Sú to notifikácie o dokončených operáciach.
- **Role-Request:** Správy ktoré sú dôležité hlavne v prípade, keď je prepínač pripojený k viacerým kontrolérom súčasne. Vďaka ním si vie kontrolér potom nastaviť správne kanál a svoju úlohu.
- **Asynchronous-Configuration:** Využíva sa k filtrovaniu asynchrónnych správ. Význam má hlavne pri pripojení prepínača k viacerým kontrolérom.[1]

2.2 Komunikácia

Protokol Openflow využíva v komunikácii medzi kontrolérom a prepínačom šifrované TLS spojenie alebo aj TCP spojenie. Spojenie inicializuje prepínač. Niektoré OpenFlow prepínače majú možnosť inicializácie spojenia z kontroléru, vtedy je ale vynútené TLS, aby sa vyhlo neautorizovanému prístupu. Kontrolér má typicky otvorených niekoľko spojení k viacerým prepínačom. Môže využívať aj niekoľko spojení k jednému prepínaču ak je potreba využívať paralelizmus z dôvodu vyššieho výkonového zaťaženia prepínača. Prepínač môže mať otvorené jedno spojenie alebo aj viaceré ak je pripojený k viacerým kontrolérom.[1]

2.3 Tabuľky tokov

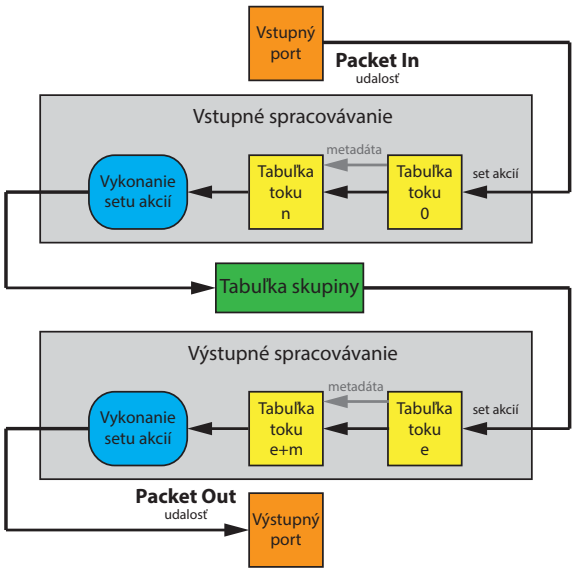
Tabuľky tokov obsahujú niekoľko položiek. Každá položka obsahuje pole:



Obr. 2.2: Zloženie záznamu toku v tabuľke tokov

- **Match fields**: Porovnávacie pole, ktoré sa skladá zo vstupného portu, hlavičky paketu, prípadne z iných polí ako vybraná hodnota inej tabuľky.
- **Priority**: Slúži pre porovnávanie priorít medzi záznamami tabuľky tokov.
- **Counters**: Toto pole sa aktualizuje ak sa paket zhoduje s pravidlom.
- **Instructions**: Slúži pre modifikáciu akčnej sady a porovnávacích procesov.
- **Timeouts**: Maximálny čas alebo aj čas nečinnosti po ktorom je pravidlo toku expirované a následne odstránené.

- **Cookie**: Vybraná hodnota kontrolérom určená k filtrovaniu záznamov toku ovplyvnenou štatistikou prietoku dát alebo požiadavkou na vymazanie alebo modifikovanie toku. Nepoužíva sa pri spracovaní rámca.
- **Flags**: Označuje zmenu spôsobu správy záznamu toku.[1]



Obr. 2.3: Spracovávanie paketov pri OpenFlow 1.5.1

2.4 Tabuľky skupín

Táto tabuľka obsahuje záznamy skupín. Každý záznam je identifikovaný pomocou ID skupiny a obsahuje tieto polia:

Group Identifier	Group Type	Counters	Action Buckets

Obr. 2.4: Zloženie záznamu skupiny v tabuľke skupiny.

- **Group Identifier:** Unikátne 32-bitové číslo identifikujúce skupinu v OF.
- **Group Type:** Slúži pre určenie sémantiky skupiny. Prepínač nemusí podporovať všetky typy skupín ale len tie ktoré sú vyžadované. Kontrolér sa dotazuje prepínača aké typy voliteľných skupín podporuje.
  - *indirect (vyžadovaná):* Spúšťa sa iba jedna skupina akcií v skupine. Umožňuje záznamom tokov alebo skupín rýchlo ukazovať na identifikátor skupiny, čo umožňuje rýchlejšiu konvergenciu. Prepínač podporujú veľké množstvo týchto skupín oproti ostatným skupinám pre jej jednoduchosť.
  - *all (vyžadovaná):* Spúšťa všetky skupiny akcií. Využíva sa pri multicastovom a broadcastovom prepínaní. Paket sa efektívne klonuje pre každú skupinovú

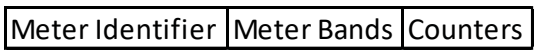
akciu, jeden paket je spracovaný pre každú skupinovú akciu v skupine. Ak je paket naklonovaný pre vstupný port, zahodí sa. Pomocou správy `OFPP_IN_PORT` je však možné nastaviť port na preposielanie paketov.

  - *select (voliteľná):* Spúšťa jednu skupinu akcií v skupine. Pakety sú spracované jedinou skupinou akcií v skupine podľa prepínacieho algoritmu (napr. hash, jednoduchý round-robin). Konfigurácia a stavy algoritmu sú externé pre OpenFlow. Vďaka algoritmom je možné rozdelenie záťaže. Pružne reaguje na výpadky portov spracovaním paketov zostávajúcimi skupinami akcií, čo pomáha redukovať výpadky linky alebo prepínača.
  - *fast failover (voliteľná):* Spúšťa prvú skupinovú akciu. Každá akčná skupina je asociovaná so špecifickým portom, môže byť aj so skupinou, ktoré udržujú ich životnosť. Skupina dovoľuje prepínaču zmeniť prepínanie bez preposlania paketu kontroléru. Pakety sa zahadzujú ak nie sú žiadne skupinové akcie živé. Skupina musí implementovať mechanizmus na udržanie životnosti skupiny akcii.

- **Counters:** Pole sa aktualizuje pri spracovaní paketu prislúchajúcou skupinou.
- **Action bucket:** Usporiadaný zoznam akčných skupín, kde každá skupina obsahuje sadu akcií k vykonaniu a asociované parametre. Vždy je aplikovaný ako sada akcií.[1]

2.5 Tabuľky merania

Tabuľky obsahujú meracie záznamy, ktoré merajú jednotlivé toky. Vďaka tomu je možné implementovať QoS operácie k prislúchajúcej šírke pásma. Merače sú nezávislé od front portov ale často sa tieto dve možnosti kombinujú pre implementáciu DiffServ. Merač aktualizuje hodnotu podľa priradených paketov a kontroluje ich množstvo. Merače sú pripojené priamo k záznamom tokov. Ak to prepínač podporuje, záznam toku môže obsahovať inštrukcie merača v sade inštrukcií. Merač obsahuje a aktualizuje hodnotu podľa všetkých záznamov ku ktorým je priradený.



Obr. 2.5: Zloženie meracieho záznamu v meracej tabuľke.

Každý merací záznam je identifikovaný na základe unikátneho meracieho identifikátora a obsahuje:

- **Meter identifier:** 32-bitové kladné číslo unikátne identifikujúce merací záznam.
- **Meter bands:** Nezoradený list meracích skupín, kde každá skupina špecifikuje hodnotu skupiny a spôsob spracovania paketu.
- **Counters:** Čítač aktualizovaný spracovaním paketov meraním.

Rôzne záznamy tokov v jednej tabuľke tokov môžu používať rovnaké meranie, iné meranie alebo aj žiadne. Pakety môžu ísť cez niekoľko meracích tabuliek a to sériovo, ale aj paralelne ak to prepínač dovoľuje.[1]

2.6 Zmeny v špecifikáciách OpenFlow

Nakoľko sa môžeme stretnúť v reálnom SDN prostredí s rôznymi verziami použitých protokolov OpenFlow, je dobré poznať zmeny ktoré boli postupne zanášané v nových špecifikáciách. Prvá verzia vznikla 28. marca 2008. Postupne sa pridávali nové funkcie, špecifikácia a popisy technológie však vznikli až pri verzii OpenFlow 0.8.9.

- **OpenFlow 0.2.0 (protokol 1)** - je prvá verzia ktorá vznikla bez špecifikácie.
- **OpenFlow 0.2.1 (protokol 1)** - modifikovaná verzia prvej verzie, ktorá vznikla ešte v ten istý deň.
- **OpenFlow 0.8.0 (protokol 0x83)** - do tejto verzie boli pridané viaceré vylepšenia z nich najdôležitejšie sú:
  - zmena typov správ,
  - pridaná priorita tokov,
  - pridané chybové správy,
  - pridané správy *Table Stats* a *Port Stats*.
- **OpenFlow 0.8.1 (protokol 0x83)** - neobsahuje žiadne zmeny od verzie 0.8.0, obsahuje len drobné opravy.
- **OpenFlow 0.8.2 (protokol 0x85)** - verzia upravuje komunikačné správy:
  - pridané správy *Echo Request* a *Echo Reply*,
  - úprava správ na 64 bitov.
- **OpenFlow 0.8.9 (protokol 0x97)** - táto verzia obsahuje prvú špecifikáciu OpenFlow switch! Pridané sú nové funkcie a správy:
  - záznamy toku musia mať masku podsiete, maska je v inverznej podobe,
  - rozšírenie *Port Stats* o viac informácií,
  - správa *In Port* dostala úpravu pre posielanie po spoločnom rozhraní čo je dôležité pri bezdrôtovej sieti,
  - správy o stave rozhrania a rozšírení pridanom výrobcom hardvéru
  - spracovávanie IP fragmentov,
  - podpora 802.1D (STP),
  - *hard timeout* záznamov,
  - pridaná správa *Hello*.
- **OpenFlow 0.9 (protokol 0x01)** - verzia priniesla tieto dôležité funkcie:
  - pridaná podpora záložného kontroléru,
  - pridané bariérové správy.
- **OpenFlow 1.0 (protokol 0x01)** - verzia prináša len drobné zmeny:
  - podpora porovnaní IP ToS/DSCP bitov,
  - *flow time* (doba prietoku) je v nanosekundách (predtým v ms).

- **OpenFlow 1.1 (protokol 0x02)** - touto verziou sa zaviedli veľmi zaujímavé funkcie a schopnosti ktoré sú veľkým prínosom k popularite OpenFlow:
  - podpora viacerých flow table (tabuliek tokov),
  - L2, L3 tabuľky,
  - podpora značkovania VLAN, MPLS (vkladanie, úprava, mazanie),
  - podpora virtuálnych portov,
  - riešenie prerušenia spojenia medzi kontrolérom a prepínačom.
- **OpenFlow 1.2 (protokol 0x03)** - verzia protokolu kedy vznikla organizácia ONF (Open Networking Foundation), ktorá prináša podporu IPv6 a iné:
  - úpravy správ *Packet In*, *Flow Mod*,
  - pribudla správa pre generovanie vlastých chybových správ.
- **OpenFlow 1.3.0 (protokol 0x04)** - verzia prišla s niekoľkými zmenami popisov portov a tabuliek a mimo iné:
  - pridanie *multipart* frameworku,
  - rozšírená podpora hlavičiek IPv6,
  - pridaná *table-miss*,
  - pridané tabuľky meraní, čítače tokov,
  - filtrovanie podľa spojenia,
  - pridané cookies do *Packe In* správ,
  - oprava chýb predchádzajúcej verzie.
- **OpenFlow 1.3.1 (protokol 0x04)** - opravené chyby z verzie 1.3.0.
- **OpenFlow 1.3.2 (protokol 0x04)** - spojenie už môže byť inicializované aj z kontroléru, ak je tak prepínač nastavený.
- **OpenFlow 1.3.3 (protokol 0x04)** - IANA vyhradila pre Openflow port 6653, čo je implementované do komunikácie.
- **OpenFlow 1.3.4 (protokol 0x04)**
  - úprava návestí pri IPv6 záznamoch tokov (pridané maskovanie),
  - oprava Push MPLS.
- **OpenFlow 1.3.5 (protokol 0x04)**
  - pridaná špecifikácia pre alternatívne transportné spojenie OpenFlow,
  - pridaná špecifikácia pre Controller Channel Connection URI zo špecifikácie OpenFlow 1.5.1,
  - zmena v číslovaní a v radení tabuliek,
  - zmena vo Flow-mod dotazoch s príkazmi *loose modify* and *loose delete*.
- **OpenFlow 1.4.0 (protokol 0x05)** - štandard bol rozšírený o mnohé popisy portov a správ.
- **OpenFlow 1.4.1 (protokol 0x05)** - úpravy synchronizácie a monitoringu:
  - pridaná synchronizačná chyba `OFPFMFC_IS_SYNC`,
  - pri synchronizácií záznamov sa záznamy prepisujú namiesto zlučovania,
  - pridanie schopností prepínačov monitorovať toky,
  - úprava synchronizácie tabuliek.

- **OpenFlow 1.5.0 (protokol 0x06)**: Prelomovejšia verzia ktorá priniesla viacero vylepšení zvyšujúcich popularitu protokolu OpenFlow:
  - pribudla *Egress* tabuľka, ktorá dokáže spracovávať pakety podľa výstupu z procesu spracovania,
  - pakety sú spracovávané už aj *Output* pravidlom (dovtedy len *Input*),
  - pribudlo spracovanie iných typov paketov, napr. protokolu IP a PPP (dovtedy bol len Ethernet),
  - úprava *Flow entry* štatistík a ich *triggeru*,
  - pridaná Copy-field akcia, Packet register, porovnávanie TCP flagov.
- **OpenFlow 1.5.1 (protokol 0x06)** - v súčasnosti posledná verzia protokolu s drobnými zmenami:
  - pridanie chyby `OFPBAC_BAD_METER` pre *bad meter* vo *flow-móde*,
  - zrušenie spôsobu mapovania paketu do *meter band*.[1].

3 SIMULÁCIA SDN A JEJ ANALÝZA.

3.1 All-In-One SDN Developer Starter

3.1.1 Inštalácia a prvé spustenie

All-in-one SDN App Development Starter VM od SDN Hub je virtuálny stroj postavený na operačnom systéme Ubuntu 14.04 64-Bit. Obsahuje niekoľko programov na testovanie či vývoj SDN sieti. Predpokladom je základná znalosť operačného systému typu Debian a virtualizačného nástroja VirtualBox.

Úspešné spustenie sa dá zhrnúť do nasledujúcich krokov:

1. Import stiahnutého .ova súboru do programu VirtualBox. Doporučené ponechať minimálne 2vCPU a 2GB RAM.
2. Pri problémoch so štartom je možné súbor rozbaliť a skúsiť použiť VMDK súbor importom do vlastného virtuálneho stroja.
3. Skontrolovať pripojenie do internetu resp. jeho nastavenie vo VirtualBoxi.
4. Použiť na prvé prihlásenie meno a heslo „ubuntu“.
5. Vytvoriť topológiu pomocou integrovaného programu Mininet.
6. Spustiť a nastaviť ľubovolný kontrolér či aplikáciu.

3.1.2 Možnosti virtuálneho stroja

Vo virtuálnom stroji sú v zložke `/home` k dispozícií predinštalované tieto projekty:

- **Mininet:** Jednoduchý simulačný nástroj siete, v ktorom je možné naprogramovať sieť ľubovolnej technológie a parametrov s podporou simulácie Open vSwitch. Softvér poskytuje podporu OpenFlow 1.3.
- **Open vSwitch:** Je multivrstvový virtuálny prepínač, licencovaný licenciou open source Apache 2.0. Bol navrhnutý na pre simuláciu v programovacom prostredí a pritom si zachoval štandardné protokoly a správu rozhraní ako bežný hardware. Podporuje protokoly ako NetFlow, sFlow, IPFIX, RSPAN, CLI, LACP, 802.1ag.[8]
- **Wireshark:** Program na kompletnú analýzu sieťovej prevádzky.
- **JDK 1.8, Eclipse Luna, Maven 3.3.3:** Vývojové prostredie a nástroje určené pre vývojárov aplikácií.
- **POX:** Mladší príbuzný NOXu (NOX je považovaný za prvý SDN kontrolér ktorý sa vyvíjal spolu s OpenFlow protokolom). Momentálne je to jeden z najviac rastúcich frameworkov pre programovanie OpenFlow kontroléru. Vyžaduje znalosť programovacieho jazyka Python.
- **RYU:** Komponentovo-orientovaný framework pre SDN siete. Obsahuje komponenty s definovaným API, vďaka ktorým je vhodný pre vývojárov aplikácií riadiacich siete. Ryu podporuje v súčasnosti viaceré protokoly na správu sieťových zariadení, napríklad OpenFlow 1.0, 1.2, 1.3, 1.4, 1.5 + Nicira Extensions, Netconf, OF-config. [5]

- **Pyrethic:** Programovací jazyk rodiny Frenetic, ktorý využíva Python. V súčasnosti sa už nepokračuje v jeho vývoji a podpore.[7]
- **Trema:** Framework na vývoj OpenFlow kontroléru v jazykoch Ruby a C.[6]
- **OpenDaylight:** Je open-source, vysoko-dostupný, modulárny, rozširovateľný, škálovateľný, multi-protokolový kontrolér naprogramovaný v jave pre SDN siete. Poskytuje užívateľom platformu na vytváranie aplikácií, fungujúcich so širokým spektrom hardvéru a protokolmi typu *south-bound*.
- **Floodlight:** OpenFlow kontrolér licencovaný Apache licenciou, založený na programovacom jazyku Java. Je podporovaný veľkou komunitou developerov, vrátane členov *Big Switch Networks*. Vývoj projektu je posledné roky spomalený.
- **ONOS:** Je skratkou *Open Network Operating System*. Je to Open Source operačný systém pre sieťové prvky. Poskytuje riadiacu rovinu pre softvérovo definované siete, spravuje prepínače a linky a riadi programy či moduly poskytujúce komunikačné služby susediacim sieťam alebo koncovým zariadeniam.[4]

3.2 Základy programu Open vSwitch

Open vSwitch je virtuálny prepínač, ktorý nám poskytuje dostatočnú náhradu za hardvér s podporou OpenFlow. Kontroléry ktoré máme k dispozícii disponujú podporou OpenFlow do verzie 1.3.0 a podporu tejto verzie nám poskytuje aj OVS.

Na základné ovládanie v simuláciách obsiahnutých v tejto práci treba poznať niektoré príkazy. Tie sa vyskytujú aj v Python skriptoch použitých pri vytváraní topológie programom Mininet.

Základné utility ktoré máme k dispozícii sú:

- **ovs-appctl:** utilita pre dotazovanie a kontrolu OVS démona,
- **ovs-vsctl:** utilita pre dotazovanie a konfiguráciu OVS démona,
- **ovs-ofctl:** utilita pre administráciu OpenFlow funkcií prepínača.

Príkaz ktorý vloží do prepínača „s1“ do tabuľky tokov pravidlo s najvyššou prioritou a s inštrukciou, aby posielal všetky pakety pre ktoré nemá definované toky do kontroléra na ďalšie spracovanie:

```
ovs-ofctl add-flow s1 -OOpenFlow13 priority=1,action=output:controller
```

Výpis všetkých záznamov tokov nachádzajúcich sa v OVS prepínači:

```
ovs-ofctl dump-flow s1 -OOpenFlow13
```

Zapnutie podpory STP protokolu na simulovanom OVS prepínači „s1“:

```
ovs-vsctl set bridge s1 stp-enable=true
```

Zapnutie STP na prepínači „s1“ a nastavenie ho ako root:

```
ovs-vsctl set bridge s1 stp-enable=true other_config:stp-priority=1
```

Zapnutie Rapid Spanning Tree protokolu (RSTP) na prepínači „s1“:

```
ovs-vsctl set bridge s1 rstp-enable=true
```

Zapnutie RSTP na prepínači „s1“ a nastavenie ho ako root:

```
ovs-vsctl set bridge s1 rstp-enable=true other_config:rstp-priority=1
```

Príkaz, ktorý zobrazí stav „Bridge“ na prepínači „s1“, kde je mimo iné možné skontrolovať aj stav STP či RSTP:

```
ovs-vsctl list bridge s1
```

Príkazy zasielané z CLI Mininetu treba posielať s príkazom sh na začiatku, napr.:

```
mininet> sh ovs-vsctl list bridge s1
```

V Python skripte sa príkazy zasielajú v tomto tvare:

```
s1.cmd('ovs-vsctl list bridge s1')
```

Samozrejme OVS nám poskytuje oveľa väčšie možnosti, ktoré ale pre svoju obsiahlosť nie sú predmetom tejto práce.

3.3 Základy programu Mininet

Používanie programu je veľmi jednoduché a stačí poznať pár základných príkazov. Prvým príkazom musíme inicializovať program Mininet, napríklad následovným príkazom, kde už priamo určíme topológiu vytvorenej siete:

```
sudo mn --topo tree,3 --mac --switch ovsk,protocols=OpenFlow13 \
    --controller=remote,ip=127.0.0.1,port=6633
```

Tento príkaz vytvorí stromovú topológiu so 7 prepínačmi, 8 počítačmi, všetky prepínače prepojí do kontroléra 127.0.0.1 na porte 6633, nastaví ich ako Open vSwitch s protokolom OpenFlow 1.3.

Parametre tohto príkazu sú:

- `--controller=remote,ip=127.0.0.1,port=6633`: Nastaví pripojenie k vzdialenému kontroléru 127.0.0.1 na porte 6633. Medzi parametrami nesmie byť medzera!!!
- `--switch ovsk,protocols=OpenFlow13`: Zapnutie emulácie Open vSwitch a nastavenie protokolu OpenFlow 1.3.
- `--mac`: Zjednoduší mac adresy počítačov na malé adresy, ktoré sú prehľadnejšie. Východzie adresy sú totiž generované náhodne.
- `--test none`: zaznamenáva čas potrebný pre vytvorenie topológie.
- `--topo tree,3`: Vytvorí jednoduchú stromovú topológiu s 3 hostiteľmi.
- `--topo single,3`: Vytvorí jednoduchú topológiu typu hviezda s 3 hostiteľmi.

Ďalšou možnosťou inicializácie programu Mininet je spustenie predkonfigurovanej siete z Python skriptu, ktorý máme vopred pripravený:

```
sudo mn --custom mojasiet.py
```

Skript je možné volať aj priamo pomocou Pythonu:

```
python ./mojasiet.py
```

Výpis sieťovej konfigurácie na hostiteľovi h1 dosiahneme použitím príkazu:

```
h1 ifconfig
```

Vypnutie a zapnutie linky medzi prepínačom s1 a koncovým bodom h1:

```
link s1 h1 down
link s1 h1 up
```

Zasiela ICMP Echo Request a ICMP Echo Reply:

```
ping 127.0.0.1
ping h1
```

„Pinganie“ z každého bodu na každý bod (veľmi užitočná funkcia ktorá naučí kontrolér polohu všetkých sieťových zariadení vďaka výmene ARP):

```
pingall
```

Otvorí nové CLI terminálu na koncovom bode h1:

```
xterm h1
```

Pred spustením virtuálnej siete programom Mininet je dobré vykonať reset nastavení, pretože prepínače zostanú nakonfigurované v databáze, ktorú treba premazať. Na toto slúži príkaz:

```
sudo mn -c
```

Ukončenie práce v programe Mininet a návrat do východzej konzoly:

```
mininet> exit
```

3.4 Analýza dostupných kontrolérov

V adresári `/home/ubuntu` sú všetky nástroje a softvér s ktorým možno pracovať. Nakoľko je cieľom práce vytvoriť jednoduchý návod pre vytvorenie si predstavy o tom ako dané riešenie približne vyzerá, je uvedený aj postup na spustenie softvéru.

Zistené poznatky z oboznámením sa s kontrolérmi môžno zhrnúť takto:

Pox: Kontrolér s podporou OpenFlow 1.0 napísaný programovacím jazykom Python. Zmeny chovania siete sa vykonávajú programovaním kódu kontroléru aplikácií. Neobsahuje žiadne grafické rozhranie.
Spustenie kontroléra:

```
cd /home/ubuntu/pox
./pox.py log.level --DEBUG forwarding.tutorial_l2_hub
```

Ryu: Kontrolér s podporou OpenFlow 1.3 napísaný v Pythone, zmeny chovania kontroléru sa priamo programujú v kóde. K dispozícií je na stiahnutie aj grafické rozhranie ktoré ale obsahuje veľa chýb, nezobrazuje správne topológiu a taktiež nemá možno vytvárania tabuliek tokov. Na disku je k dispozícií aplikácia ktorá vykonáva jednoduché L2 prepínanie.
Spustenie kontroléra:

```
cd ~/ubuntu/ryu
./bin/ryu-manager --verbose ryu/app/simple_switch_13.py
```

Opendaylight: Program obsahuje množstvo funkcií ktoré sa dajú pridávať a odoberať, dostupné GUI zobrazuje asi najlepšie topológiu siete ktorú kontrolér spravuje. Nedá sa žiaľ cez GUI pridávať záznamy tokov a ani ich editovať. K tomu je určené YANG-UI API čo je pomerne zložité bez aplikácie ktorá ho dokáže použiť.
Spustenie kontroléra:

```
cd ~/SDNHub_Opendaylight_Tutorial/distribution/opendaylight-karaf/target/\
    assembly
./bin/karaf
```

Floodlight: K dispozícii je na disku kontrolér verzie 1.0. Tento kontrolér obsahuje podporu Openflow 1.0. Posledná vydaná verzia ktorá podporuje túto verziu a a navyše experimentálne aj verziu OpenFlow 1.4 sa volá Floodlight 1.2. Kontrolér je napísaný programovacím jazykom Java a každú zmenu, vrátane konfigurácie, je treba kompilovať. Obsahuje vlastné GUI, kde je možné zobraziť spustené moduly, prepínače a ich tabuľky tokov.
Spustenie kontroléra:

```
cd ~/floodlight
java -jar ./target/floodlight.jar
```

ONOS: Správa tohto kontroléru je v terminálovom okne. Podobne ako u kontroléru OpenDaylight sa dajú príkazmi pridávať a odoberať funkcie. Riešenie obsahuje veľa príkazov a aplikácií pre správu siete. Do kontroléra je možné doinštalovať GUI. ONOS podporuje klástrovanie pri viacerých kontroléroch čo je jeho najväčšou výhodou.
Spustenie kontroléra:

```
cd ~/onos
source ./tools/dev/bash_profile
karaf clean
```

4 POROVNANIE VÝKONNOSTI SDN KONTROLÉROV.

Cieľom prvého merania bolo zistiť rozdiely výkonnosti jednotlivých kontrolérov ktoré sú k dispozícií na disku. Testované kontroléry boli aktualizované na poslednú verziu z GitHub repozitára a postupné spúštané v upravených konfiguráciach, s ktorými bolo možné ich použiť spolu s testovacími nástrojmi. Ako testovací nástroj bol dostupný *Cbench*, *MT-Cbench* (multivláknový Cbench) a *WTCbench* (MT-Cbench s rozšírením vykresľovania grafov). Prvé dva z nich boli použité na všetky kontroléry a výsledky merania boli následne porovnané.

Popis hardware a zapojenia:
Na meranie bol pôvodne použitý 1 x server HP Proliant DL360e Gen8. Na tomto serveri bežal kontrolér a aj testovací nástroj. K preťaženiu procesoru však dochádzalo už pri testoch s malým počtov prepínačov (> 50), čoho následkom boli časté havárie aplikácie určenej na testovanie výkonnosti kontroléru. Namerané výsledky mali taktiež veľký rozptyl. Toto riešenie sa pre účely merania bolo nedostačujúc presné, preto bolo zvolené riešenie s dvomi servermi na ktorých bežal separátne testovací nástroj a kontrolér.
Výsledná konfigurácia testovacieho hardware bola:

Kontrolér:
HP Proliant DL360e Gen8
8GB RAM
Intel(R) Xeon(R) CPU E5-2603 v3 @ 1.60GHz (4 jadrá)

Testovací nástroj:
HP Proliantl DL80 Gen9
2GB RAM
Intel(R) Xeon(R) CPU E5-2603 v3 @ 1.60GHz (6 jadier)

Na oboch serveroch bežal virtualizačný server Microsoft Hyper-V 2012 R2. Virtuálny stroj ktorý bol k dispozícií na meranie bol vo formáte VirtualBox a preto bol prevedený na formát .vhd ktorý bolo následne možné použiť na týchto serveroch. Na testovací stroj boli následne doinštalované testovacie utility Cbench a MT-Cbench.

4.1 Cbench

Tento nástroj vznikol ako prvý nástroj na testovanie SDN kontrolérov. Jeho hlavnou nevýhodou je jednovláknové riešenie. Pri väčších zaťaženiach s viacerými prepínačmi Cbench zaťažoval jednovláknovo kontrolér, ktorý tak doviedol k havárií. V reálnom prostredí by však každý prepínač tvoril samostatné vlákno v kontroléri, preto si Cbench zachováva presnosť len pri meraniach s menším počtom prepínačov, kedy kontrolér zvláda dané vlákno.

Tento nástroj je možné jednoducho nainštalovať podľa návodu na stránke projektu:[13]

```
sudo apt-get install autoconf automake libtool libsnmp-dev libpcap-dev
git clone git://gitosis.stanford.edu/oflops.git
cd oflops; git submodule init && git submodule update
git clone git://gitosis.stanford.edu/openflow.git
cd openflow; git checkout -b release/1.0.0 remotes/origin/release/1.0.0
wget http://hyperrealm.com/libconfig/libconfig-1.4.9.tar.gz
tar -xvzf libconfig-1.4.9.tar.gz
cd libconfig-1.4.9
./configure
sudo make && sudo make install
cd ../../netfpga-packet-generator-c-library/
sudo ./autogen.sh && sudo ./configure && sudo make
cd ..
sh ./boot.sh ; ./configure --with-openflow-src-dir=<absolute path to \
    openflow branch>; make
sudo make install
cd cbench
```

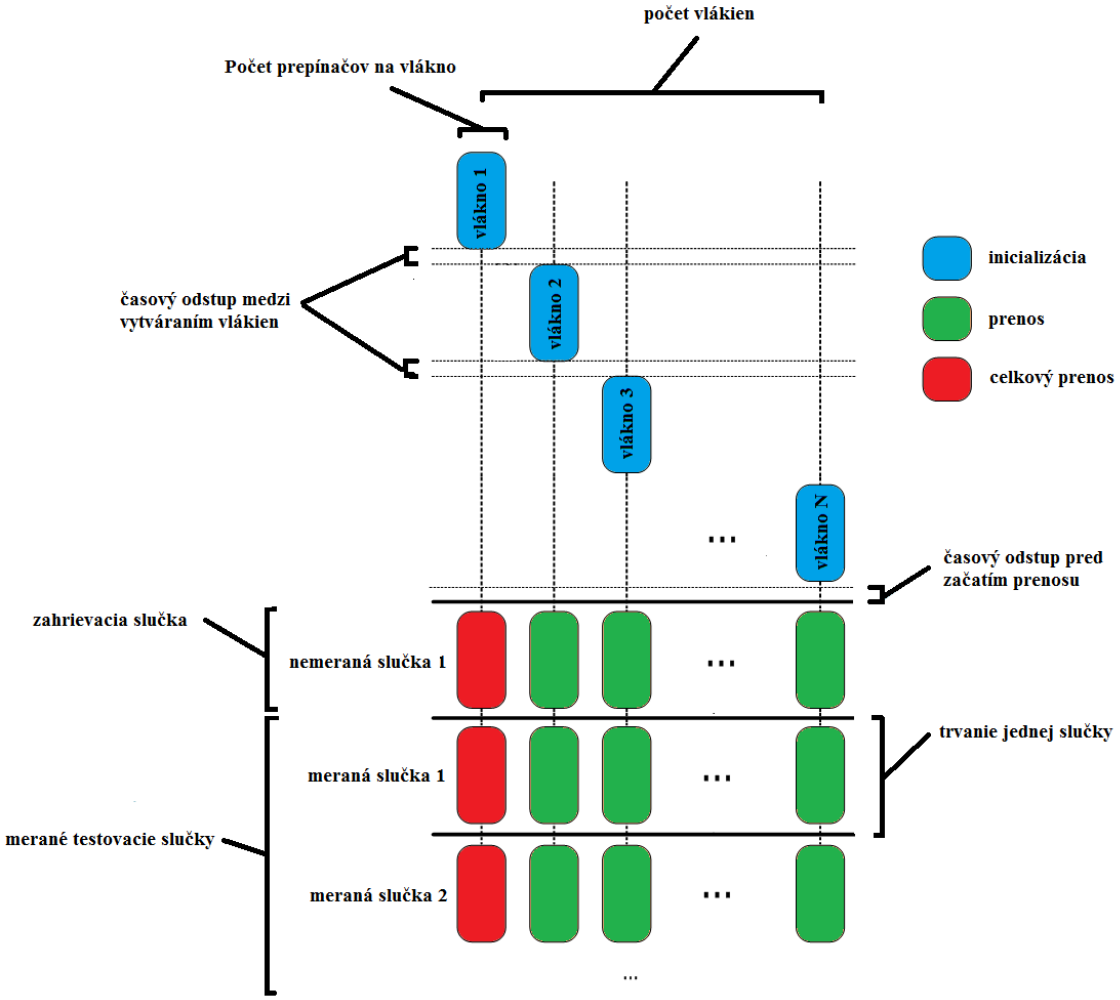
Argumenty s ktorými je cbench možné spúšťať sú:

- -c : server hostujúci SDN kontrolér
- -p : port kontroléru (6633 alebo 6653)
- -m : počet milisekúnd medzi meraniami
- -D : pauza pred prvým meraním v ms
- -l : počet opakovaní merania
- -s : maximálny počet prepínačov
- -M : počet hostov (MAC adries) na jednom prepínači
- -r : stupňovité meranie 1...n počtu prepínačov[13]

4.2 MT-Cbench

Tento nástroj vznikol ako rozšírená verzia nástroja Cbench, ktorá používa viaceré vlákna pre generovanie OpenFlow dátového toku z mnohých paralelných streamov. Účelom bolo vytvoriť lepší nástroj ako Cbench, ktorý dokáže otestovať topológiu siete vytvorenú kontrolérom OpenDaylight. Cbench totiž pri väčšom počte prepínačov ako 200 často spôsoboval haváriu kontroléru a výsledky meraní boli preto veľmi nepresné. MT-Cbench nie je potrebné inštalovať ani kompilovať, stačí stiahnuť predkompilovaný repozitár a spustiť z neho pripravený skript.

```
git clone https://github.com/intracom-telecom-sdn/mtcbench.git
cd mtcbench
./run.sh
```



Obr. 4.1: MT-Cbench architektúra.

Dôležité parametre dostupné MT-Cbench sú:

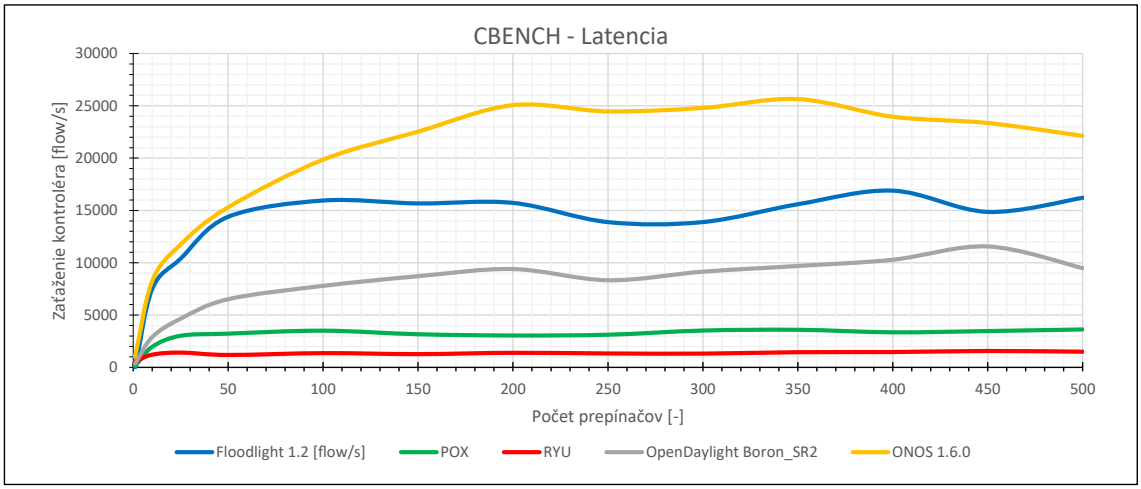
- -Z : celkový počet MT-Cbench vlákien
- -S : celkový počet prepínačov na jedno vlákno
- -T : oneskorenie medzi vytvorením vlákien
- -D : oneskorenie začatia prevádzky siete
- -c : ip adresa kontroléra
- -p : port kontroléra
- -M : počet unikátnych harvérových adries na jeden prepínač
- -l : počet iterácií na jeden test
- -m : trvanie jednej iterácie
- -t : zapnutie módu priepustnosti
- -w : počet počiatočných iterácií, ktoré nie sú zarátané do merania
- -c : počet konvcových iterácií, ktoré nie sú zarátané do merania

4.3 Latencia kontrolérov

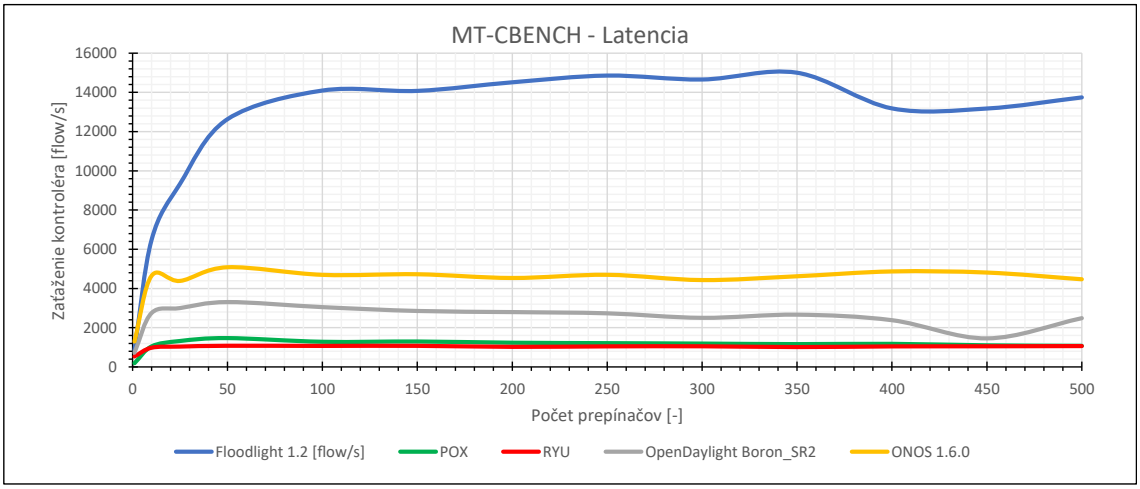
Latenciu u kontroléra môžeme vyjadriť ako čas potrebný k spracovaniu jedného paketu. V režime latencie sa na kontrolér vyšle v časovom intervale učené množstvo požiadaviek za sebou. Každá ďalšia požiadavka sa zasiela až po obdržaní odpovede z predchádzajúcej požiadavky. Takto sa meria koľko príde odpovedí za sekundu. Výsledný počet flow/s je možné obrátením previesť na čas potrebný k vybaveniu jednej flow.

Merania boli vykonávané nasledujúcim príkazom:

```
1 cbench -c controller_ip -p 6653 -m 1000 -D 1000 -l 60 -s 500 -M 1000 -r
```



Obr. 4.2: Meranie latencie kontrolérov - Cbench.



Obr. 4.3: Meranie latencie kontrolérov - MT-Cbench.

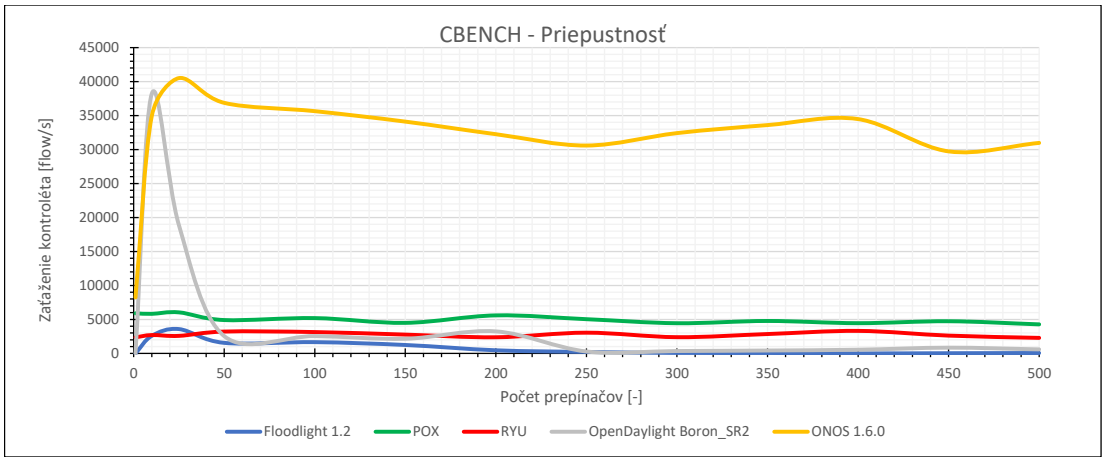
Z nameraných hodnôt pomocou Cbench vidieť že všetky kontroléry zanášajú nižšiu latenciu pri väčšom počte prepínačov. Pri MT-Cbench je už tento vývoj merania iný, hodnoty sú prakticky ustálené. Pravdepodobne je to dané tým, že kontrolér nie je stavaný na jednovláknový stream kontrolných dát. Meranie pomocou MT-Cbench by malo reálnejšie odzrkadlovať skutočné maximálne hodnoty flow/s ktoré dokáže kontrolér vybaviť. V teste latencie dosiahli najlepšie výsledky Floodlight kontrolér vo verzii 1.2 s latenciou pod 70 µs a ONOS kontrolér s odozvami pod 200 µs. K dobrým výsledkom sa dopracoval aj OpenDaylight s latenciou do 290 µs avšak s veľmi veľkou chybovosťou merania, ktorá vznikala vyťažením kontroléra z MT-Cbench.

4.4 Priepustnosť kontrolérov

Nasledujúcim testom merania priepustnosti bolo zisťované, koľko `Packet_IN` správ dokáže kontrolér spracovať za sekundu. Na kontrolér sa vyšle v časovom intervale obrovské množstvo požiadaviek a meria sa počet odpovedí z kontroléra. Na meranie bol taktiež použitý nástroj Cbench a MT-Cbench. U oboch boli výsledky veľmi podobné až na Opendaylight kontrolér. MT-Cbench primárne určený na meranie Opendaylight vykazoval veľmi veľkú chybovosť a rozptyl merania a preto sú výsledky málo vierohodné. Cbench pri väčšom počte prepínačov nedokázal merať kontrolér vôbec a meral nulové hodnoty.

Merania boli vykonávané nasledujúcim príkazom:

```
cbench -c controller_ip -p 6653 -m 1000 -D 1000 -l 60 -s 500 -M 1000 -r -t
```



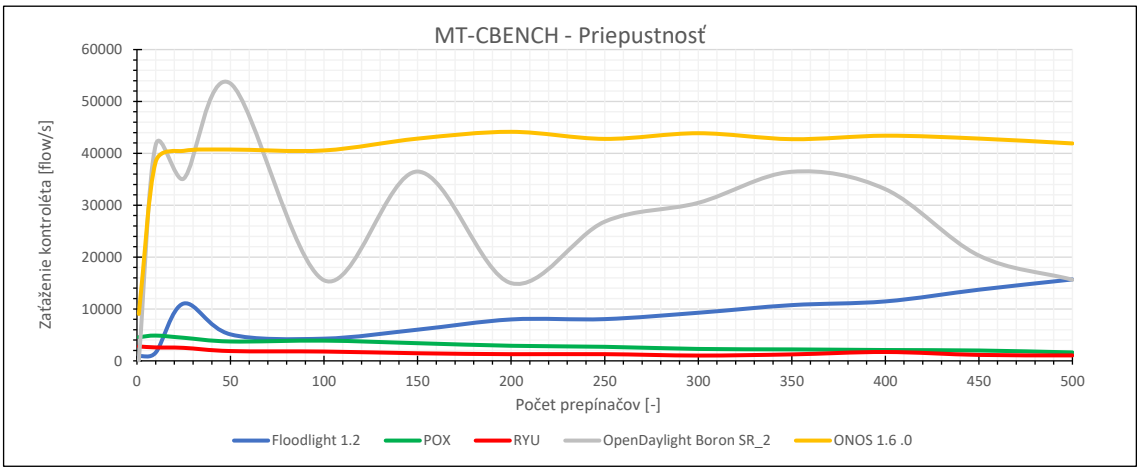
Obr. 4.4: Meranie priepustnosti kontrolérov - Cbench.

Merania boli vykonávané nasledujúcim predpripraveným bash skriptom:

```
cd ./mtcbench
./run.sh SDN-SERVER 6633 500 1 500 1000 1000 1000 65 1000 5 Throughput
```

Tento skript používa parametre MT-Cbench v poradí ako je zrejmé z prvotného výpisu po štarte:

```
Starting CBENCH with: CONTROLLER_IP:SDN-SERVER CONTROLLER_PORT:6633 \
    GENERATOR_THREADS:150 GENERATOR_SWITCHES_PER_THREAD:1 \
    GENERATOR_SWITCHES:500 GENERATOR_THREAD_CREATION_DELAY_MS:1000 \
    GENERATOR_DELAY_BEFORE_TRAFFIC_MS:1000 GENERATOR_MS_PER_TEST:1000 \
    GENERATOR_INTERNAL_REPEATS:10 GENERATOR_MACS:1000 GENERATOR_WARMUP:0 \
    GENERATOR_MODE:Throughput
cbench: controller benchmarking tool
   running in mode 'throughput'
   connecting to controller at SDN-SERVER:6633
   faking 500 switches with 500 threads :: 65 tests each; 1000 ms per test
   with 1000 unique source MACs per switch
   learning destination mac addresses before the test
   starting test with 1000 ms delay after features_reply
   ignoring first 5 "warmup" and last 0 "cooldown" loops
   debugging info is off
```



Obr. 4.5: Meranie priepustnosti kontrolérov - MT-Cbench.

Meranie ukázalo že rôzne kontroléry dosahujú takmer zhodnú priepustnosť v priemere do 5000 flow/s. Výnimkou bol len ONOS kontrolér, ktorý sa výrazne líšil od ostatných priepustnosťou do 40 000 flow/s, pri meraní nástrojom MT-Cbench dokonca do 50 000 flow/s. Nástroj MT-Cbench sa ukázal ako vhodnejší pri meraní kontroléru Floodlight, kde boli pri väčšom počte prepínačov namerané reálnejšie hodnoty s menším rozptylom merania. Z nameraných hodnôt je vidieť, že kontroléry POX a RYU narozdiel od ostatných, dokážu udržať nameranú priepustnosť aj pri nízkom počte prepínačov (<25).

Počet prepínačov [–]		1	10	25	50	100	150	200	250	300	350	400	450	500
Floodlight 1.2 [flow/s]	cbench	0	7492	10397	14394	15948	15670	15720	17676	13890	15592	16892	14861	16207
Floodlight 1.2 [flow/s]	mt-cbench	857	6480	9339	12635	14094	14077	14515	14858	14658	15003	13184	13178	13744
POX [flow/s]	cbench	129	1967	3021	3227	3507	3166	3048	3121	3520	3591	3352	3472	3625
POX [flow/s]	mt-cbench	188	1045	1322	1470	1282	1296	1234	1212	1189	1162	1176	1100	1074
RYU [flow/s]	cbench	547	1213	1413	1187	1361	1267	1393	1330	1315	1451	1464	1567	1496
RYU [flow/s]	mt-cbench	534	979	1037	1078	1071	1075	1023	1048	1052	1012	1046	1052	1063
OpenDaylight [flow/s]	cbench	356	2888	4658	6519	7790	8715	9398	8328	9148	9694	10287	11560	9487
OpenDaylight [flow/s]	mt-cbench	726	2749	3001	3308	3049	2853	2796	2732	2503	2665	2383	1455	2485
ONOS 1.6.0 [flow/s]	cbench	1047	8253	11754	15303	19855	22536	25064	24466	24801	25643	23960	23357	22129
ONOS 1.6.0 [flow/s]	mt-cbench	1293	4647	4386	5083	4695	4725	4536	4698	4429	4623	4869	4812	4469

Tab. 4.1: Meranie latencie kontrolérov pomocou Cbench a MT-Cbench

Počet prepínačov [–]		1	10	25	50	100	150	200	250	300	350	400	450	500
Floodlight 1.2 [flow/s]	cbench	0	2580	3580	1550	1658	1241	459	212	72	62	3	29	58
Floodlight 1.2 [flow/s]	mt-cbench	902	1587	11049	5101	4276	6009	7982	8044	9271	10742	11455	13712	15598
POX [flow/s]	cbench	5902	5823	6040	4908	5196	4495	5594	5035	4444	4774	4448	4735	4271
POX [flow/s]	mt-cbench	4564	4872	4446	3733	3898	3401	2905	2706	2295	2208	2087	1992	1624
RYU [flow/s]	cbench	2366	2700	2580	3213	3140	2764	2374	3048	2388	2849	3315	2618	2285
RYU [flow/s]	mt-cbench	2774	2585	2502	1887	1795	1461	1288	1289	1022	1244	1706	1144	1047
OpenDaylight [flow/s]	cbench	3212	6845	2834	355	0	0	0	0	0	0	0	0	0
OpenDaylight [flow/s]	mt-cbench	90	41588	35124	53457	15525	36487	14980	26825	30455	36445	33090	20298	15634
ONOS 1.6.0 [flow/s]	cbench	8229	34928	40516	36869	35647	34110	32259	30591	32417	3360	34483	29730	30974
ONOS 1.6.0 [flow/s]	mt-cbench	9035	38350	40478	40727	40556	42835	44147	42783	43876	42727	43396	42846	41904

Tab. 4.2: Meranie priepustnosti kontrolérov pomocou Cbench a MT-Cbench

4.5 Záver z merania

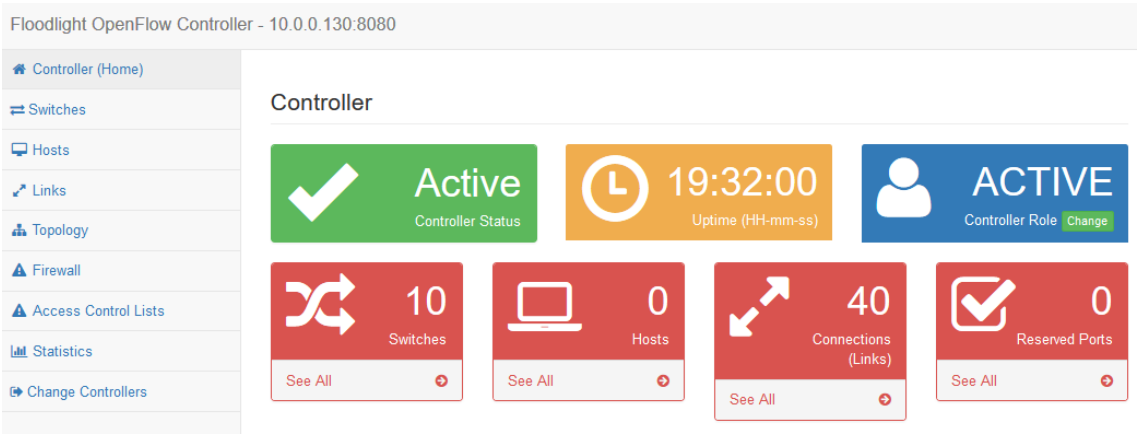
Tento model merania predpokladá najhoršiu situáciu v sieti ktorá sa prakticky nevyskytuje. Je to situácia, kedy je každý prepínač vyťažený rovnomerne a dátové toky sú generované z tisícov smerov súčasne. Obrovské množstvo prepínačov sa navyše vyskytuje len v dátových centrách. Zo získaných výsledkov je možné si približne nadimenzovať počet kontrolérov a ich výkon na predpokladanú záťaž v sieti. Typickú prevádzku by mal zvládnuť každý testovaný kontrolér, čo však nie je možné overiť v simulovanom prostredí s dostupnými nástrojmi.

5 MERANIA NA KONTROLÉROCH

5.1 FLOODLIGHT

Z meraných kontrolérov bol na dôkladnejšiu analýzu vybraný ako prvý kontrolér, ktorý mal stabilné výsledky merania, bohatú dokumentáciu, použiteľné grafické prostredie a zaujímavé základné aplikácie. Týmto kontrolérom bol Floodlight vo verzií 1.2. Ten poskytuje zaujímavé moduly, ktoré sú jeho súčasťou, a prehľadnosť kódu umožňuje jednoducho zaniesť zmeny do konfigurácie a teda chovania kontroléru.

Grafické rozhranie je momentálne dostupné v projekte *Floodlight-WebGUI*[1].



Obr. 5.1: Webové rozhranie kontroléru Floodlight.

Vďaka jednoduchosti nájdenia a rozbehnutia komplexného a funkčného riešenia sa podarilo vytvoriť systém, ktorý cez grafické prostredie umožňuje používať najdôležitejšie funkcie kontroléru a to zobraziť topológiu siete, prepínače a ich vlastnosti, tabuľky tokov. Navyše ako jedno z mála voľne dostupných riešení tu funguje aj pridávanie a modifikovanie prietokových pravidiel. Toto ucelené riešenie poskytne východiskové prostredie pre rôzne merania v tejto práci.

5.1.1 Inštalácia a spustenie kontroléru

K správnemu fungovaniu webového rozhrania je potrebné stiahnuť verziu z repozitára v ktorom sú zanesené najnovšie zmeny. Opravný kód ktorý je k dispozícií pre Mikrotik funguje len s touto verziou.

```
cd ~/
git clone https://github.com/rizard/floodlight.git
cd floodlight-master
java -jar ./target/floodlight.jar
```

[1] https://github.com/floodlight/floodlight-webui

5.1.2 Inštalácia webového rozhrania

Najaktuálnejšiu verziu možno stiahnúť do projektu floodlight z repozitára[2] následovne:

```
cd floodlight
git pull origin master
git submodule init
git submodule update
```

5.1.3 Pripojenie simulovanej siete do kontroléra

Na skúšku funkčnosti pripojenia nasledujúci príkaz vytvorí sieť s jedným prepínačom, piatimi počítačmi a jedným kontrolérom. Ku kontroléru sa pripojí prepínač protokolom OpenFlow 1.3.

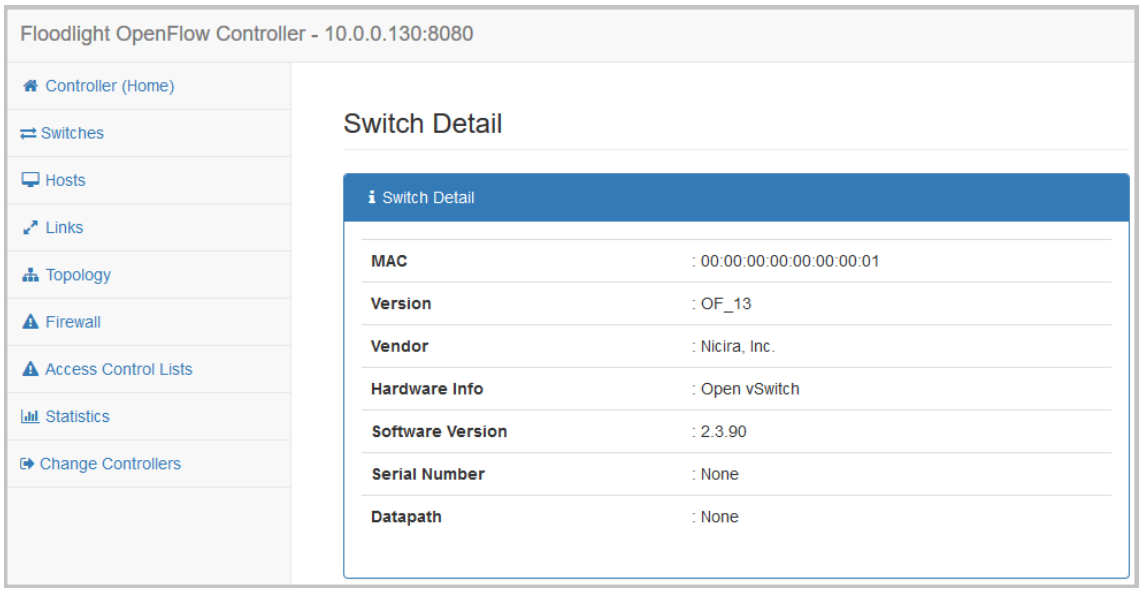
```
sudo mn --topo single,5 --mac --switch ovsk,protocols=OpenFlow13 --\
    controller remote,ip=127.0.0.1,port=6653
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1) (h5, s1)
*** Configuring hosts
h1 h2 h3 h4 h5
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```

Keďže Floodlight kontrolér vo východzom nastavení automaticky nainštaluje pravidlo ktoré posiela všetky neznáme pakety kontroléru a kontrolér automaticky inštaluje tabuľky tokov podľa paketov prijatých `Packet_In` správami, stačí pre funkčnosť zaslať ICMP echo request programom ping. Ak funguje všetko správne nasledujú odpovede:

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5
h2 -> h1 h3 h4 h5
h3 -> h1 h2 h4 h5
h4 -> h1 h2 h3 h5
h5 -> h1 h2 h3 h4
*** Results: 0% dropped (20/20 received)
```

[2] https://github.com/rizard/floodlight

Overiť funkčnosť GUI je možné otvorením webovej adresy v nejakom prehliadači:
`http://SDN-SERVER/pages/switchDetail.html?macAddress=00:00:00:00:00:00:00:01`
V sekcií „Switches“ je možné vidieť všetky pripojené prepínače, verziu OpenFlow ktorou komunikujú a tabuľky tokov ktoré sú v prepínači nainštalované. Tabuľky tokov miznú po 5 sekundách čo spôsobuje konfigurácia kontroléru na IdleTimeout = 5s. Ak v tomto čase neprejde žiadny paket tabuľkou so zodpovedajúcimi parametrami, tabuľka sa vymaže.



Obr. 5.2: Zobrazenie podrobností pripojeného virtuálneho prepínača.

Topológiu vytvorenej virtuálnej siete možno nájsť tu:
`http://SDN-SERVER/pages/topology.html`
Pomocou mapy topológie je možné nabehnutím myši na vybraný prepínač dynamicky zobrazovať tabuľky tokov, ktoré aktuálne využíva. Vďaka tomu je možné jednoducho skúmať cestu, ktorou paket prechádza. Táto funkcia pomôže pri meraní jedného z parametrov siete a tým je konvergencia.

5.1.4 Meranie výkonnosti modulov

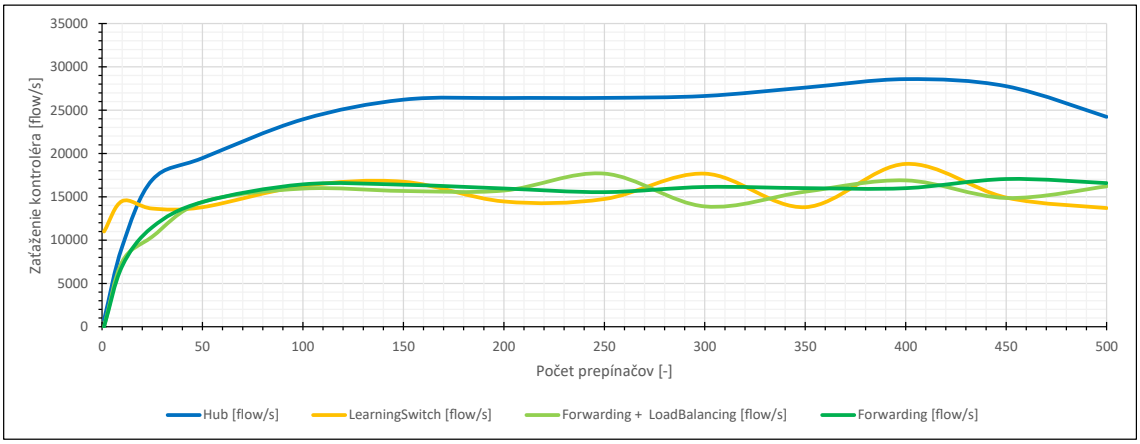
Kontrolér floodlight obsahuje tieto moduly umožňujúce doručovanie paketov:

- **Hub:** Táko aplikácia, ktorá je vo východzom nastavení vypnutá šíri, záplavovou akýkoľvek prijatý paket na všetky ostatné aktívne porty.
- **LearningSwitch:** Aplikácia sa správa ako bežný L2 prepínač. Vytvorí rozhranie REST API pre čítanie informácií z tabuľky prepínača (vlany, porty, hosty)
- **Forwarding:** Východzia aplikácia ktorá vykonáva reaktívne prepínanie. Podporuje len určité topológie.
- **LoadBalancing:** Aplikácia ktorá umožňuje rozloženie záťaže pre ICMP, TCP a UDP dátové toky.

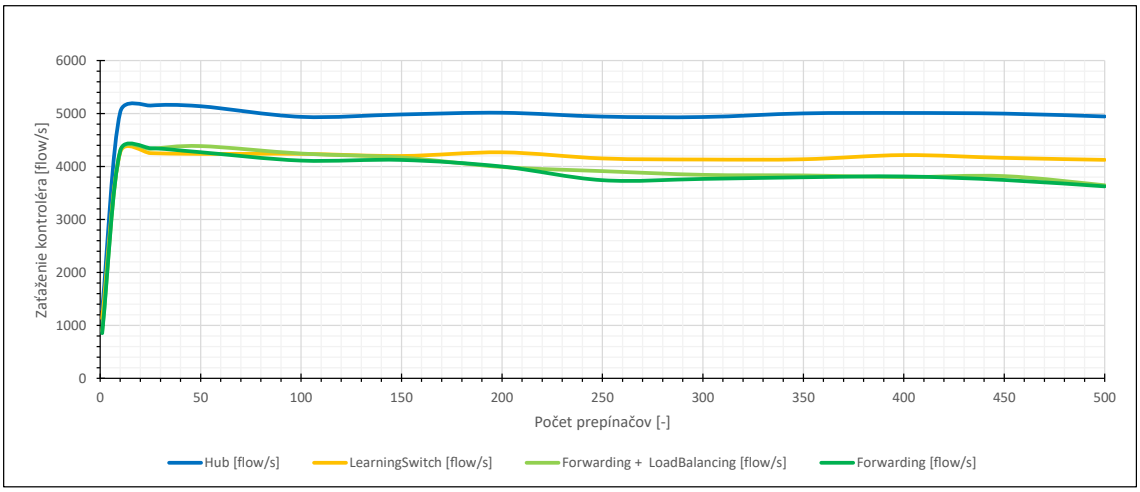
- **Firewall:** Aplikácia umožňujúca ACL definovanie pravidiel podľa zadaných kritérií. Pomocou REST API je možné aplikáciu vypínať či zapínať a pridávať, mazať či zobrazovať aktuálne nastavené pravidlá.
- **PortDownReconciliation:** Aplikácia reaguje na výpadky portov či liniek v sieti tým, že preprogramuje toky v sieti.
- **StaticFlowEntryPusher:** Pomocou tejto aplikácie je možné staticky nastavovať tabuľky tokov v OpenFlow prepínačoch.
- **Virtual Network Filter:** Aplikácia izolácie siete pracujúcej podľa MAC adries.

Prvé 4 moduly boli zvolené k záťažovému testu podobne ako pri meraní výkonnosti kontrolérov v predchádzajúcej kapitole. Opäť boli použité nástroje Cbench a MT-Cbench.

Pri meraní latencie boli hodnoty prakticky identické:

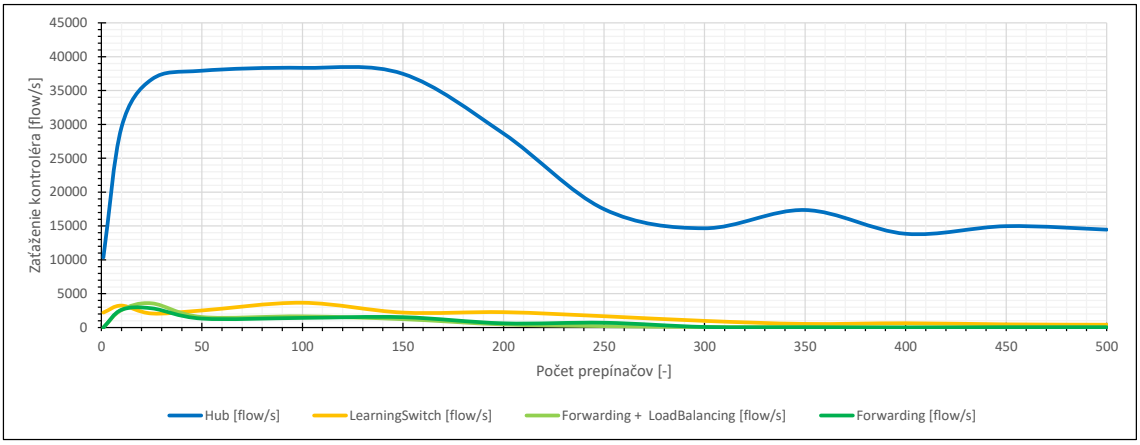


Obr. 5.3: Floodlight - Latencia modulov podľa Cbench

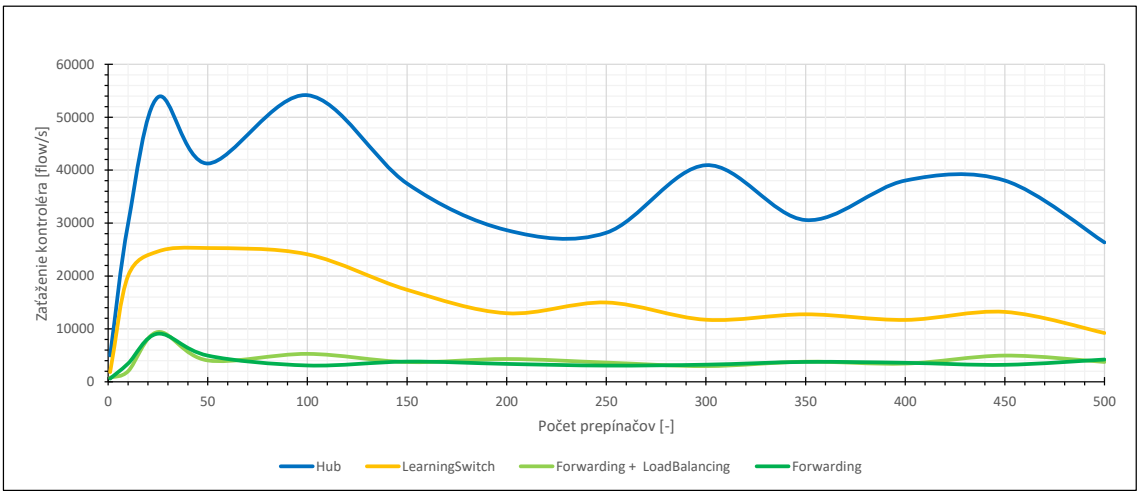


Obr. 5.4: Floodlight - Latencia modulov podľa MT-Cbench

Pri meraní priepustnosti už bola viditeľnejšia nestabilita merania nástrojom MT-Cbench:



Obr. 5.5: Floodlight - Priepustnosť modulov podľa Cbench



Obr. 5.6: Floodlight - Priepustnosť modulov podľa MT-Cbench

Z výsledkov je zrejmé že aplikácie, ktoré boli testované nemali veľký vplyv na latenciu a priepustnosť kontrolérov okrem aplikácie Hub. Nakoľko aplikácia Hub nepotrebuje poznať sieť pretože len záplavou všetko šíri na aktívne porty, je jej logika veľmi jednoduchá a preto dokáže vyriešiť naprogramovanie tokov dát veľmi rýchlo. Najzaujímavejší modul Forwarding, ktorý dokáže inteligentne spravovať sieť pomocou ďalších aplikácií ako LoadBalancig a PortDownReconciliation pri danej hardvérovej konfigurácií zvládal priemerne 5000 flow/s pri latencii do 0.4 ms, čo je priemer medzi SDN kontrolérmi.

Počet prepínačov [–]		1	10	25	50	100	150	200	250	300	350	400	450	500
Forwarding + LB [flow/s]	cbench	0	7492	10397	14394	15948	15670	15720	17676	13890	15592	16892	14861	16207
Forwarding + LB [flow/s]	mt-cbench	890	4287	4332	4385	4244	4169	3990	3912	3842	3831	3800	3818	3645
Forwarding [flow/s]	cbench	0	6987	11489	14401	16434	16391	15959	15537	16135	16000	15993	17051	16578
Forwarding [flow/s]	mt-cbench	852	4299	4345	4269	4111	4123	3999	3741	3765	3796	3812	3745	3624
Hub [flow/s]	cbench	789	9228	16953	19471	23941	26212	26399	26414	26636	27605	28582	27766	24226
Hub [flow/s]	mt-cbench	1299	5028	5150	5137	4937	4982	5014	4942	4934	5002	5010	4998	4944
LearningSwitch [flow/s]	cbench	10989	14499	13642	13799	16250	16739	14461	14743	17672	13802	18794	14911	13698
LearningSwitch [flow/s]	mt-cbench	1130	4268	4252	4237	4238	4198	4267	4152	4131	4136	4215	4162	4125

Tab. 5.1: Meranie latencie kontrolérov

Počet prepínačov [–]		1	10	25	50	100	150	200	250	300	350	400	450	500
Forwarding + LB [flow/s]	cbench	0	2580	3580	1550	1658	1241	459	212	72	62	3	29	58
Forwarding + LB [flow/s]	mt-cbench	781	1995	9415	4046	5283	3736	4311	3636	2964	3801	3444	4959	3750
Forwarding [flow/s]	cbench	0	2621	2845	1328	1426	1536	625	712	86	42	15	45	42
Forwarding [flow/s]	mt-cbench	609	3449	9094	4953	3092	3823	3385	3068	3240	3762	3595	3212	4215
Hub [flow/s]	cbench	10350	29559	36649	37936	38356	37471	28655	17490	14664	17361	13850	14977	14460
Hub [flow/s]	mt-cbench	4977	29843	53787	41257	54164	37471	28655	28174	40935	30570	38032	38032	26339
LearningSwitch [flow/s]	cbench	2211	3241	2080	2522	3663	2213	2273	1672	984	538	639	472	422
LearningSwitch [flow/s]	mt-cbench	1828	20015	24635	25278	24096	17390	12959	14988	11731	12756	11695	13214	9221

Tab. 5.2: Meranie priepustnosti kontrolérov

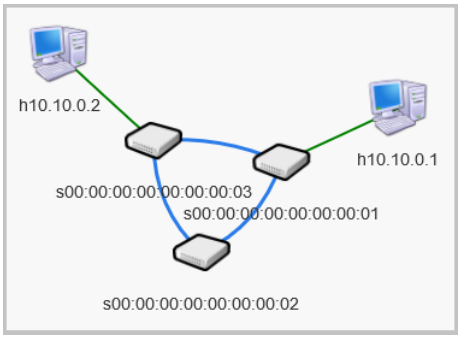
5.1.5 Konvergencia pri poruche linky

Jedným z prvých parametrov, ktoré boli zvolené na meranie, je konvergencia siete pri výpadku linky v MASH topológii. Jedna zo základných funkcií OpenFlow kontroléru je vytvorenie najkratšej cesty. Nakoľko má kontrolér k dispozícií všetky informácie o sieti priamo z prepínačov, mal by vedieť vypočítať a nastaviť novú cestu v krátkom čase.

Ako prvé je potrebné zistiť, či sa vie kontrolér vo východzej konfigurácií vysporiadať so slučkami v sieti. Poslúži na to skript generujúci topológiu s dvomi počítačmi a tromi prepínačmi, pričom prepínače sú navzájom prepojené. Na simuláciu takejto siete je už potreba vytvoriť skript v Pythone. V tomto skripte je zvolená latencia na každej linke na 0.25 ms pre jednoduchšiu analýzu komunikácie. Tá nemá vplyv na rýchlosť konvergencie pretože prepínače komunikujú s kontrolérom bez obmedzení po virtuálnom kanáli.
Skript k-nostp-n3.py A.1 spustíme nasledovným spôsobom:

```
sudo python k-nostp-n3.py
```

Po spustení siete a použití programu `pingall` je vidieť, že všetko funguje. GUI kontroléru ukazuje že pakety prúdia jednou obojsmernou cestou:



Obr. 5.7: Sieť vytvorená skriptom k-nostp-n3.py

Nasledovným príkazom je možné overiť priamo na prepínačoch že je tomu tak:

```
sudo ovs-ofctl dump-flows s2
```

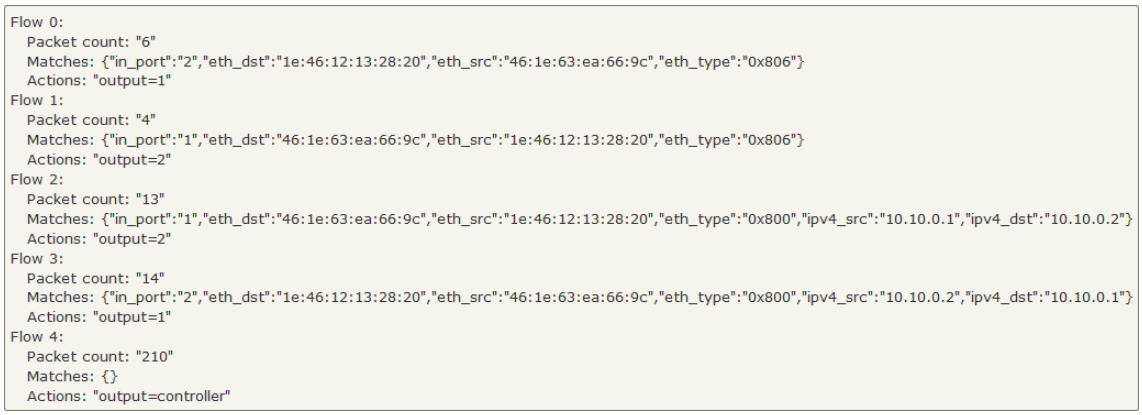
Kontrolér teda zvláda slučky v sieti, vypočíta najkratšiu cestu medzi prepínačmi čo zodpovedá udávanej vlastnosti „Forwarding“ modulu kontroléra Floodlight.

Postup merania

Otvorením terminálového okno príkazom `xterm h1` je možné zasielať ICMP Echo Request priamo z h1 na h2.

```
ping 10.10.0.2
```

Z mapy topológie je možné nájsť linku, cez ktorú prebieha tok dát. Nabehnutím myši na prepínač v sekcií „Topology“ vyskočí okno, kde je možné tieto údaje vyčítať:



Obr. 5.8: Zobrazenie naprogramovaných záznamov tokov na prepínači cez WEBGUI

Po spustení zachytávania dát cez Wireshark možno linku vypnúť príkazom:

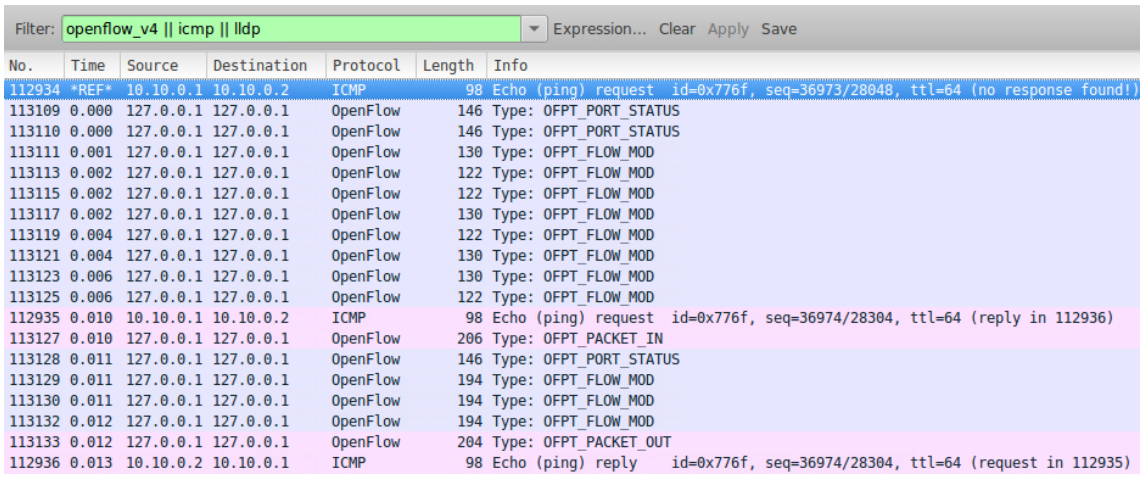
```
mininet> link s1 s3 down
```

Po zastavení zachytávania paketov a vyfiltrovaní Openflow a ICMP protokolu bolo možné vyčítať potrebné informácie.

Kontrolér informoval o nasledovných udalostiach:

```
2016-12-07 12:34:04.24 INFO [n.f.l.i.LinkDiscoveryManager] Inter-switch \
    link removed: Link [src=00:00:00:00:00:00:00:01 outPort=2, dst\
    =00:00:00:00:00:00:00:03, inPort=2, latency=2]
2016-12-07 12:34:04.24 INFO [n.f.l.i.LinkDiscoveryManager] Inter-switch \
    link removed: Link [src=00:00:00:00:00:00:00:03 outPort=2, dst\
    =00:00:00:00:00:00:00:01, inPort=2, latency=1]
2016-12-07 12:34:04.29 INFO [n.f.t.TopologyManager] Recomputing topology \
    due to: link-discovery-updates
```

Sú to informácie o tom, že kontrolér obdržal LLDP správy výpadku danej linky a portov. Tieto správy spracovala aplikácia `LinkDiscoveryManager` a následne ďalšia aplikácia `TopologyManager`, ktorá pozná celú sieť, prepočítala nové programovanie siete.



Obr. 5.9: Analýza siete pri výpadku spojenia v SDN

Z analýzy paketov programom Wireshark a notifikačných správ v terminálovom okne kontroléra možno vyčítať nasledovné udalosti:

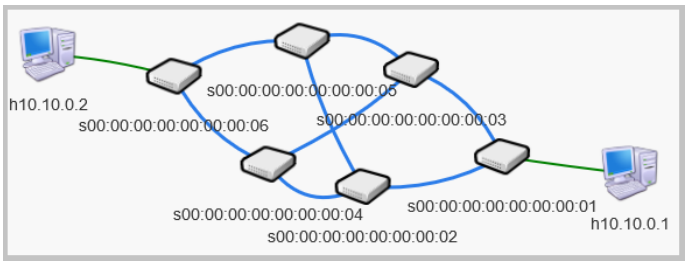
- **0.000 s** - ICMP echo request z h1 na h2
- **0.000 s** - prepínače posielajú kontroléru zmenu stavu portov na `down`
- **0.000 s** - kontrolér informuje o prijatí LLDP správ z prepínačov o prerušení spojenia
- **0.001 s** - kontrolér maže záznamy tokov ktoré súvisia s nefunkčným portom na prepínači (6ms)
- **0.006 s** - kontrolér informuje o prepočítavaní topológie
- **0.011 s** - nový ICMP echo request z h1 na h2
- **0.012 s** - kontrolér inštaluje nové záznamy tokov (2ms)
- **0.013 s** - ICMP echo reply z h2 na h1 na nový ICMP echo request

Z daných informácií je zrejmé, že čas konvergencie siete je 12 ms. Tento postup analýzy bol opakovaný 10-krát a to aj na ďalších skriptoch:

k-nostp-n6.py A.2, **k-nostp-n10.py** A.3, **k-nostp-n14.py** A.4.

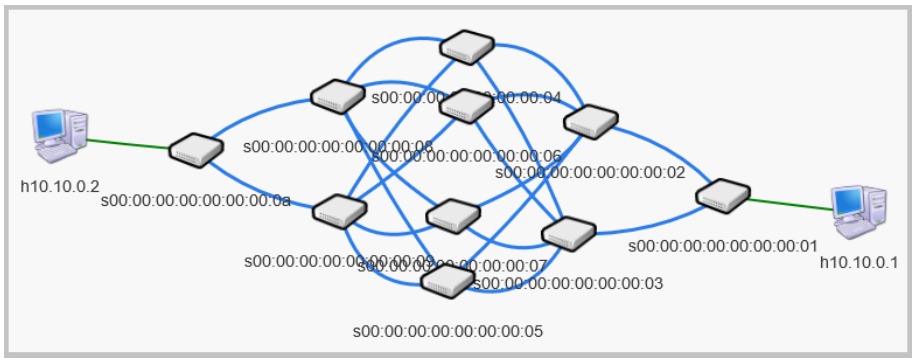
Z týchto skriptov je vytvorená zložitejšia sieť prepínačov, cez ktoré pri výpadku linky musí kontrolér prepočítať novú najkratšiu cestu.

Topológia týchto simulovaných sieti z dostupných skriptov vyzerá nasledovne:
k-nostp-n6.py: A.2 2 počítače, 6 prepínačov, 10 liniek, ICMP odozvy=2ms



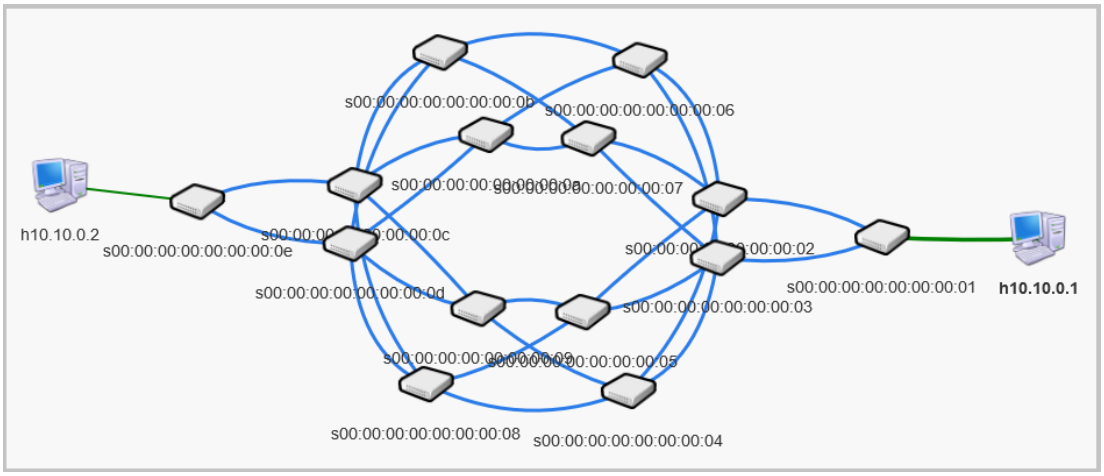
Obr. 5.10: Sieť vytvorená skriptom k-nostp-n6.py

k-nostp-n10.py: A.3 2 počítače, 10 prepínačov, 22 liniek, ICMP odozvy=3ms



Obr. 5.11: Sieť vytvorená skriptom k-nostp-n10.py

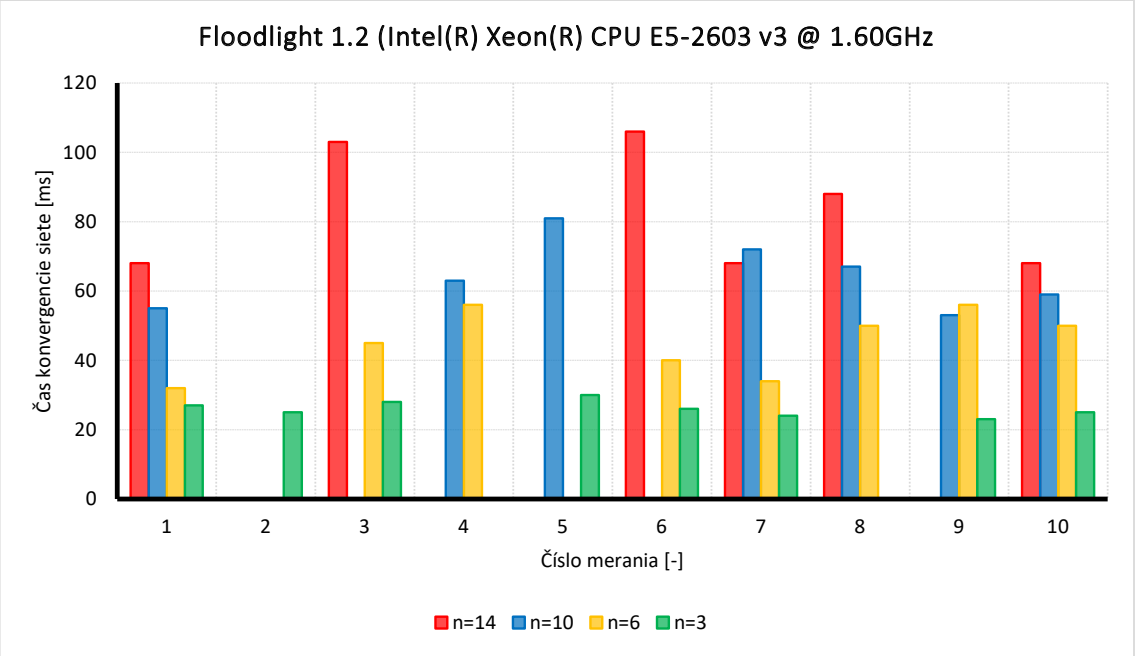
k-nostp-n14.py: A.4 2 počítače, 14 prepínačov, 30 liniek, ICMP odozvy=4ms



Obr. 5.12: Sieť vytvorená skriptom k-nostp-n14.py

Výsledky merania

Výsledné merania sú spracované do tabuľky a grafu, z ktorých je zrejmé, že rýchlosť konvergencie sa prakticky nemení s pribúdaním prepínačov v sieti. Rýchlosť je však veľmi vysoká a neklesá výrazne ani v sieťovom diametri rovnajúcemu sa 7 skokov. Počas merania však sieť niekoľkokrát nezkonvergovala, čo je pravdepodobne spôsobené chybou v aplikácií Forwarding a nie chybou technológie OpenFlow. Meranie s kontrolérom ONOS v nasledujúcej kapitole totiž tento problém prakticky nemalo.



Obr. 5.13: Konvergencia siete kontroléru Floodlight pre n počet prepínačov

Číslo merania [–]	1	2	3	4	5	6	7	8	9	10
$K_{n=14}$ [ms]	68	x	103	x	x	106	68	88	x	68
$K_{n=10}$ [ms]	55	x	x	63	81	x	72	67	53	59
$K_{n=6}$ [ms]	32	x	45	56	x	40	34	50	56	50
$K_{n=3}$ [ms]	12	25	28	x	30	26	24	x	23	25

Tab. 5.3: Konvergencia SDN siete K pre n počet prepínačov v sieti

Záver z merania

Konvergencia siete s použitým protokolom OpenFlow 1.3 je veľmi rýchla, avšak s použitím kontroléru Floodlight často nespoľahlivá. Rovnaká chyba sa vyskytovala pri testovaní všetkých verzií. Kontrolér obsahuje modul zvaný `FlowCacheReconcileManager`, ktorý má riešiť zmenu stavu linky a portu[10], avšak na spoľahlivosť konvergencie to vplyv nemalo. Taktiež je v konfigurácií nastavený parameter `remove-flows-on-link-or-port-down=TRUE`. Kontrolér však vôbec neinformuje o mazaní záznamu toku z tabuľky. Hardvér Mikrotik, ktorý bol k dispozícií na porovnanie so simulovaným prostredím disponoval iba protokolom OpenFlow 1.0 na čo kontrolér upozorňuje správou, že nie je možné zmazať tabuľky tokov s použitím tohto protokolu. Nakoľko sú kontrolér a tieto nové moduly, ktoré sa starajú o prepočítanie a nastavenie siete stále vo vývoji, je pravdepodobné, že je v nich zanesená chyba v kóde, na ktorú je náhodne možné naraziť volenou metódou merania. Z analýzy tabuliek na prepínačoch je zrejmé, že stav zablokovania siete nastáva v momente, kedy je zasielaním dát udržaný nevyexpriovaný `Idle Timeout` záznamu, vďaka čomu prepínač preposiela pakety na port, ktorý je nefunkčný. Stačí teda pozastaviť posielanie dát na čas väčší ako je `Idle Timeout`, záznam sa zmaže z prepínačov ktoré blokujú prenos a kontrolér správne preprogramuje sieť. Druhou možnosťou je nastavenie `Hard Timeout`, ktorý vymaže záznam toku periodicky vždy po uplynutí času. Zanáša tak do parametrov siete zvýšenú latenciu pri prvom prenesenom pakete po každom prepočítaní siete. Oba parametre sú súčasťou konfigurácie kontroléru. [10]

5.2 ONOS

Ďalší z kontrolérov, ktorý bol zvolený na meranie, je kontrolér ONOS. Taktiež bol vybraný na základe prepracovanej dokumentácie, výbornému grafickému prostrediu a množstvu dostupných aplikácií. Narozdiel od kontroléra Floodlight, pre ONOS existuje len jedna aplikácia na prepínanie paketov v sieti OpenFlow. Navyše nie je dostupná aplikácia na load-balancing prevádzky v dátovej rovine. ONOS však poskytuje možnosť load-balancingu master kontroléru, vďaka čomu možno záťaž rozložiť na viacero kontrolérov.

Aplikácie, ktoré potrebujeme pre prevádzku siete OpenFlow sú:

- **onos-drivers:** Ovladače k rôznym zariadeniam vrátane virtuálneho prepínača OpenVSwitch použitého programom Mininet.
- **onos-openflow:** Meta aplikácia využívajúca južné rozhranie vytvorené pre protokol OpenFlow.
- **onos-openflow-base:** Poskytovanie južného rozhrania pre protokol OpenFlow.
- **onos-app-fwd:** Aplikácia vykonávajúca reaktívne preposielanie paketov.
- **onos-lldp-provider:** Aplikácia vyhľadávania liniek prostredníctvom LLDP.
- **onos-host-provider:** Aplikácia vykonávajúca vyhľadávanie hostov v sieti.

5.2.1 Inštalácia a spustenie kontroléru

Pôvodná verzia ONOS kontroléru ktorá sa nachádzala na disku nefungovala podľa návodu, preto je potrebné stiahnúť verziu, ku ktorej fungujú aj príkazy podľa wiki na stránke projektu ONOS. Touto verziou bol ONOS 1.6.0.

```
sudo mkdir /opt
cd /opt
sudo wget -c http://downloads.onosproject.org/release/onos-1.6.0.tar.gz
sudo tar xzf onos-1.6.0.tar.gz
sudo mv onos-1.6.0.tar.gz onos
```

Spustenie kontroléru sa potom vykonáva následovne:

```
/opt/onos/bin/onos-service start
```

Do kontroléra je potrebné nainštalovať vyššie spomínané moduly pre použitie s protokolom OpenFlow v programe Mininet:

```
onos> feature:install onos-drivers
onos> feature:install onos-openflow
onos> feature:install onos-openflow-base
onos> feature:install onos-app-fwd
onos> feature:install onos-lldp-provider
onos> feature:install onos-host-provider
```

5.2.2 Inštalácia webového rozhrania

Na doinštalovanie aplikácie webového rozhrania stačí doinštalovať jeden balíček:

```
onos> feature:install onos-gui
```

Otestovanie rozhrania kontroléra je možné prihlásením sa pomocou URL:

`http://SDN-SERVER:8181/onos/ui/login.html`

Východzími prihlasovacími údajmi sú **onos/rocks**.

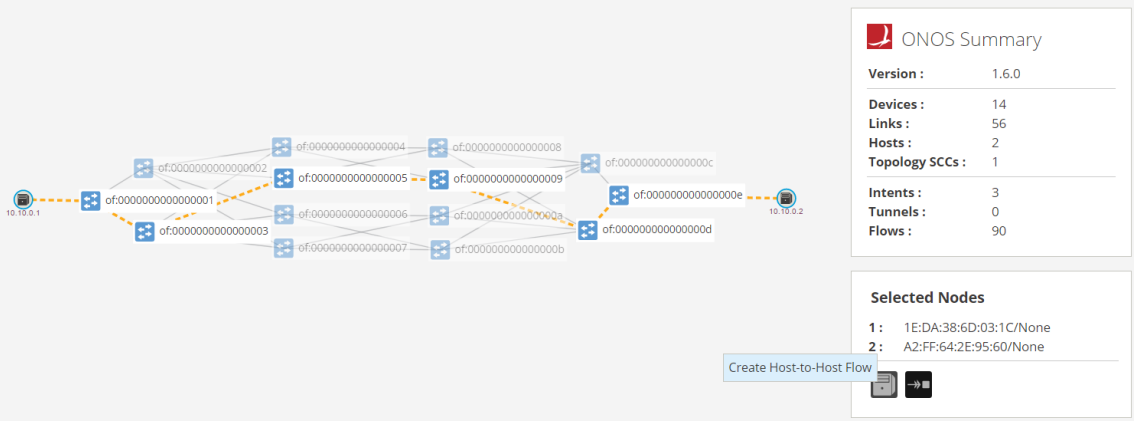
Obr. 5.14: Prihlasovacia obrazovka ONOS kontroléru.

5.2.3 Konvergencia pri poruche linky alebo portu

Podobne ako pri kontroléri Floodlight, skripty ukázali že sa sieť riadená kontrolérom ONOS dokáže bezproblémovo vysporiadať so slučkami v sieti. Kontrolér vždy vypočítal najlepšiu cestu na základe zistených počtov skokov a celkovej latencie. Z analýzy zachytených paketov a kontroly flow tabuliek vidieť, že ONOS nepoužíva parametre `HardTimeout` a `IdleTimeout`, resp. využíva ich s hodnotou nastavenou na 0. Spolieha sa vyslovene na dobrú komunikáciu s OF prepínačom, mazanie neplatných nastavení flow tabuliek a inštaláciu nových pravidiel v správnom časovom slede pomocou `BARRIER_REQUEST` správ.

Postup merania

Vďaka zobrazovaniu tokov vybraných hostov umožňuje ONOS jednoduchú identifikáciu cesty a teda používaných liniek.



Obr. 5.15: ONOS - zobrazenie toku v sieti.

Rozhranie kontroléru dokáže narozdiel od kontroléru Floodlight využívať aj meracie tabuľky vďaka ktorým vie ONOS zobrazovať rýchlosti a objem prenášaných dát cez konkrétnu linku. Taktiež nám ONOS umožňuje zobraziť aktuálne nastavenia každého prepínača, vďaka čomu je možné presne pozorovať ako sa kontrolér chová v danej situácií. Pri výpadku linky bolo preto možné občas pozorovať duplicitné zavedenie niektorých flow záznamov, toto však z princípu fungovania siete nemá vplyv na jej funkčnosť.

Spôsobom zhodným pri meraní konvergencie kontroléru Floodlight, stačí vypnúť niektorú z aktívnych liniek. Pritom sú pomocou programu Wireshark zaznamenávané všetky pakety a vyfiltrovaná komunikácia Openflow a pakety ARP a ICMP protokolu.

Konvergencia siete začína prvou udalosťou `OFP_PORT_STATUS`, kedy dostane kontrolér informáciu o prerušení spojenia, a končí poslednou udalosťou `OFP_FLOW_MOD`, kedy kontrolér nainštaluje posledné pravidlá do príslušných prepínačov, prípadne odpoveďou `OFP_BARRIER_REPLY` na dotaz `OFP_BARRIER_REQUEST` z kontroléra, ktorá oznamuje kontroléru úspešné spracovanie všetkých predchádzajúcich nastavení.

Filter: openflow_v4 || icmp Expression... Clear Apply Save

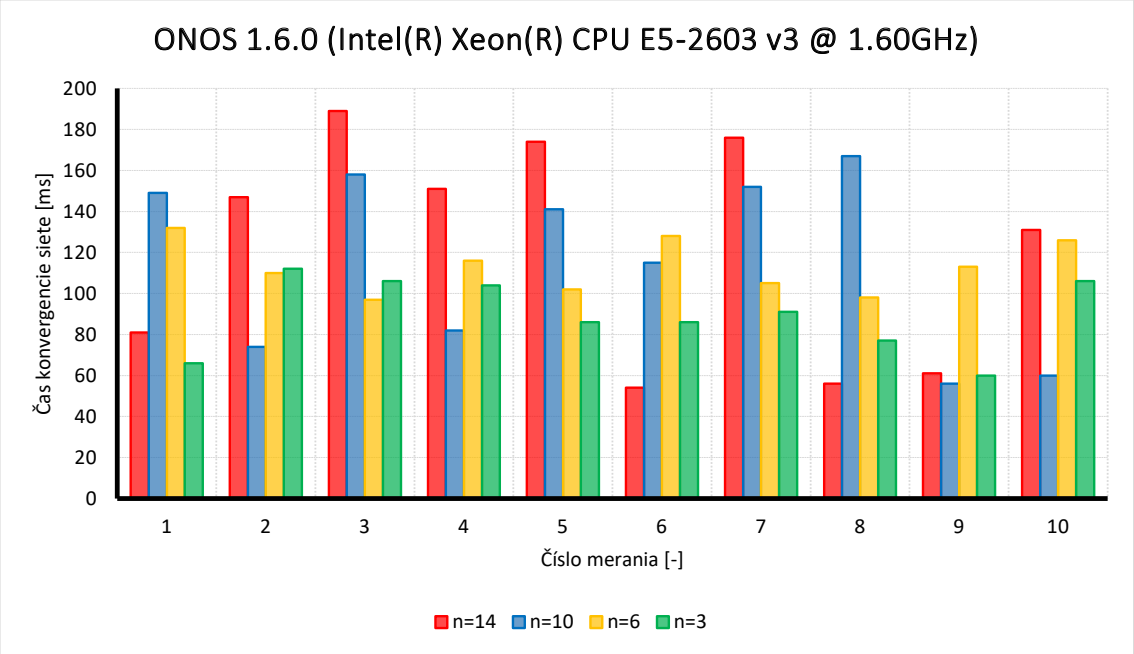
No.	Time	Source	Destination	Protocol	Length	Info
100398	2.813827	10.0.5.201	10.0.5.200	OpenFlow	146	Type: OFPT_PORT_STATUS
100399	2.813934	10.0.5.201	10.0.5.200	OpenFlow	146	Type: OFPT_PORT_STATUS
100401	2.818755	10.0.5.201	10.0.5.200	OpenFlow	146	Type: OFPT_PORT_STATUS
97933	2.831873	10.10.0.1	10.10.0.2	ICMP	98	Echo (ping) request id=0x8c00, seq=53199/53199, ttl=64 (no response found!)
97934	2.847944	10.10.0.1	10.10.0.2	ICMP	98	Echo (ping) request id=0x8c00, seq=53200/53455, ttl=64 (no response found!)
97935	2.867921	10.10.0.1	10.10.0.2	ICMP	98	Echo (ping) request id=0x8c00, seq=53201/53711, ttl=64 (no response found!)
97936	2.883957	10.10.0.1	10.10.0.2	ICMP	98	Echo (ping) request id=0x8c00, seq=53202/53967, ttl=64 (no response found!)
97937	2.899915	10.10.0.1	10.10.0.2	ICMP	98	Echo (ping) request id=0x8c00, seq=53203/54223, ttl=64 (no response found!)
100405	2.900946	10.0.5.200	10.0.5.201	OpenFlow	146	Type: OFPT_FLOW_MOD
100406	2.900948	10.0.5.200	10.0.5.201	OpenFlow	146	Type: OFPT_FLOW_MOD
100407	2.900950	10.0.5.200	10.0.5.201	OpenFlow	74	Type: OFPT_BARRIER_REQUEST
100408	2.900952	10.0.5.200	10.0.5.201	OpenFlow	146	Type: OFPT_FLOW_MOD
100409	2.900954	10.0.5.200	10.0.5.201	OpenFlow	146	Type: OFPT_FLOW_MOD
100410	2.900956	10.0.5.200	10.0.5.201	OpenFlow	74	Type: OFPT_BARRIER_REQUEST
100411	2.901240	10.0.5.201	10.0.5.200	OpenFlow	146	Type: OFPT_FLOW_REMOVED
100412	2.901394	10.0.5.201	10.0.5.200	OpenFlow	146	Type: OFPT_FLOW_REMOVED
100413	2.901453	10.0.5.201	10.0.5.200	OpenFlow	74	Type: OFPT_BARRIER_REPLY
100414	2.901544	10.0.5.201	10.0.5.200	OpenFlow	146	Type: OFPT_FLOW_REMOVED
100415	2.901594	10.0.5.201	10.0.5.200	OpenFlow	146	Type: OFPT_FLOW_REMOVED
100416	2.901721	10.0.5.201	10.0.5.200	OpenFlow	74	Type: OFPT_BARRIER_REPLY
100423	2.904799	10.0.5.200	10.0.5.201	OpenFlow	170	Type: OFPT_FLOW_MOD
100424	2.904801	10.0.5.200	10.0.5.201	OpenFlow	170	Type: OFPT_FLOW_MOD
100425	2.904803	10.0.5.200	10.0.5.201	OpenFlow	74	Type: OFPT_BARRIER_REQUEST
100427	2.905069	10.0.5.201	10.0.5.200	OpenFlow	74	Type: OFPT_BARRIER_REPLY
100428	2.905467	10.0.5.200	10.0.5.201	OpenFlow	170	Type: OFPT_FLOW_MOD
100429	2.905469	10.0.5.200	10.0.5.201	OpenFlow	170	Type: OFPT_FLOW_MOD
100430	2.905471	10.0.5.200	10.0.5.201	OpenFlow	74	Type: OFPT_BARRIER_REQUEST
100432	2.906033	10.0.5.201	10.0.5.200	OpenFlow	74	Type: OFPT_BARRIER_REPLY
100433	2.906374	10.0.5.200	10.0.5.201	OpenFlow	170	Type: OFPT_FLOW_MOD
100434	2.906377	10.0.5.200	10.0.5.201	OpenFlow	170	Type: OFPT_FLOW_MOD
100435	2.906378	10.0.5.200	10.0.5.201	OpenFlow	74	Type: OFPT_BARRIER_REQUEST
100437	2.906926	10.0.5.201	10.0.5.200	OpenFlow	74	Type: OFPT_BARRIER_REPLY
97938	2.919926	10.10.0.1	10.10.0.2	ICMP	98	Echo (ping) request id=0x8c00, seq=53204/54479, ttl=64 (reply in 97939)
97939	2.920572	10.10.0.2	10.10.0.1	ICMP	98	Echo (ping) reply id=0x8c00, seq=53204/54479, ttl=64 (request in 97938)

Obr. 5.16: ONOS - zachytené pakety pri výpadku linky

Z rozdielu časov zachytených paketov je vypočítaný čas a údaje sú zanesené do grafu 5.17.

Výsledky merania

Z nameraných hodnôt je vidieť že konvergencia siete sa drží v rozmezé 50 ms až 150 ms nezávisle na zložitosti siete. Sieť vždy zkonvergovala a kontrolér vždy vypočítal najkratšiu cestu. Zo zachytávania paketov je zrejmé že kontrolér často používa správy `BARRIER_REQUEST` a `FLOW_REMOVED` narozdiel od kontroléru Floodlight. To spôsobuje dlhší čas konvergencie ale pravdepodobne zaručuje spoľahlivejšie programovanie siete.



Obr. 5.17: Konvergencia siete s kontrolérom ONOS.

Číslo merania [–]	1	2	3	4	5	6	7	8	9	10
$K_{n=14}$ [ms]	81	147	189	151	174	54	176	56	61	131
$K_{n=10}$ [ms]	149	74	158	82	141	115	152	167	56	60
$K_{n=6}$ [ms]	132	110	97	116	102	128	105	98	113	126
$K_{n=3}$ [ms]	66	112	106	104	86	86	91	77	60	106

Tab. 5.4: Konvergencia SDN siete K pre n počet prepínačov v sieti s kontrolérom ONOS

Záver z merania

Z meraní konvergencie siete pri výpadku linky vychádzajú podobné hodnoty ako u kontroléru Floodlight, avšak chybovosť konvergencie je v tomto prípade nulová. Kontrolér sa javil ako veľmi spoľahlivý a GUI kontroléru reaguje na zmeny v sieti prakticky v reálnom čase. Kontrolér navyše podporuje fungovanie v klástri, ktoré nemá vplyv na rýchlosť reakcie zmeny dátovej cesty v prípade poruchy siete, čo spoľahlivosť siete znásobuje.

6 ZÁVER

Cieľom tejto práce bolo oboznámiť sa s architektúrou softvérovo definovaných sietí a s jej fungovaním a spravovaním. V teoretickej časti preto bola vysvetlená architektúra a delenie, objasnené výhody aj nevýhody plynúce z využívania tejto technológie, vysvetlené základy protokolu OpenFlow, ktorý je považovaný za základný prvok týchto sieti. Preto je pre OpenFlow v práci načrtnuté jeho využívanie prvkami siete a jeho architektúra ktorá je obsiahnutá v špecifikácií tohto protokolu.

Nasimulovaním siete v programe Mininet a naprogramovaním kontroléru bolo dokázané, že je možné jednoducho a hromadne spravovať viaceré prvky siete a pritom zmerať chovanie siete v určitých podmienkach. Zo simulácií je zrejmé že spustiť sieť typu SDN je vlastne veľmi jednoduché a je len na správcovi siete a jeho schopnostiach a potrebách sieť nastaviť tak ako potrebuje, či už nastavením statických pravidiel alebo použitím aplikácií ktoré riadia kontrolér cez API. Správca siete ma v súčasnosti na výber niekoľko riešení, ktoré sú Open-Source, ale taktiež riešenia s uzavretým kódom, ktoré sú spravidla platené ale zato disponujú spoľahlivejšími aplikáciami. Z Open-Source riešení, ktoré boli testované v tejto práci sa však oplatí vyskúšať nasadenie riešenia ONOS do reálnej prevádzky. Veľmi zaujímavým riešením do budúcna vyzerá byť OpenDaylight a Floodlight, ktoré zatiaľ počas merania trpeli na nestabilitu či náhodné havárie. Tieto projekty dostávajú zatiaľ pravidelné aktualizácie a skôr či neskôr sa dopracujú k stabilite. Ako jednoduché kontroléry ale zato použiteľne na základné riadenie siete vo forme jednoduchého prepínania paketov sa javili POX a RYU, trpia však na slabšiu dokumentáciu a zložitejšie ovládanie.

Medzi dôležité parametre siete patrí nesporne priepustnosť a latencia siete. Preto boli tieto parametre zmerané pomocou nástrojov Cbench a MT-Cbench. Z výsledkov vidieť že vplyv technológie OpenFlow na latenciu a priepustnosť nastáva len pri vzniku nového spojenia medzi novými koncovými bodmi a pri zmene ich umiestnenia, kedy musí kontrolér preprogramovať sieťové prvky. Pri meraní priepustnosti preto namerané výsledky ukazujú najvyťaženejší stav siete a to ten, kedy všetky koncové body v sieti začnú naraz komunikovať a vyťažia všetky prepínače na maximum a teda aj rovnomerne. Tento stav však v reálnej prevádzke nenastáva. Nástroje MT-Cbench a Cbench nedokážu nasimulovať reálnejšie modely siete. V meraní týchto parametrov sa ukázal nástroj MT-Cbench ako vhodnejší o čom svedčia stabilnejšie namerané výsledky spôsobené multivláknovým meraním. Pomocou týchto nástrojov je teda možné predbežne nadimenzovať počet kontrolérov a ich výkon na predpokladanú záťaž siete.

Ďalším parametrom siete ktorý bol meraný na dvoch vybraných kontroléroch (ONOS. Floodlight) s funkčným GUI rozhraním, bolo meranie konvergencie pri zmenách na linkách či portoch. V tomto meraní sa ukázal ako lepší práve ONOS, ktorý zaručoval konvergenciu vždy narozdiel od kontroléru Floodlight. ONOS vykazoval vysoké využívanie `BARRIER` správ, čo predchádzalo zablokovaniu siete pri programovaní. Floodlight si dobre poradil s inými topológiami siete okrem testovanej MESH topológie a narozdiel od ONOSu disponuje rozdeľovaním záťaže, kedy rôznym koncovým bodom vie prideľovať rôzne cesty.

Namerané časy boli v desiatkach milisekúnd čo zaručuje vysokú spoľahlivosť siete. Konvergencia je narozdiel od protokolu STP oveľa rýchlejšia a v porovnaní s protokolom RSTP nedochádza k blokovaniu niektorých liniek.

Práca bola zameraná na testovanie parametrov Open-Source kontrolérov, ktoré boli k dispozícií na disku virtuálneho stroja. Ako sa ukázalo, veľa z týchto projektov je stále vo vývoji a častokrát je v nich vidieť rôzne chyby a zmeny architektúry pri vydaní novej verzie. Vývoj je treba neustále sledovať pretože rýchlosť vývoja projektov sa niekedy veľmi mení. Dôkazom je napríklad kontrolér Floodlight ktorý dostáva za posledné roky len drobné opravy či úpravy a to predovšetkým v grafickom rozhraní. Napriek tomu, budúcnosť SDN sietí je pravdepodobne veľká pretože sa o ne zaujíma čoraz viac poskytovateľov internetových služieb, veľkých firiem ale hlavne dátových centier, kde je veľmi veľkou výhodou spravovať všetko dynamicky z jedného unifikovaného zariadenia. Google prezentoval využívane vlastného SDN riešenia v ich WAN sieti ako riešenia ktoré im zvýšilo priepustnosť siete.[14] Taktiež sa o SDN zaujíma napríklad Facebook pri vývoji vlastného riešenia [16] alebo aj Microsoft, ktorý využíva SDN na monitorovanie a analýzu siete. [15] Práve Open-Source riešenia, ktoré sú k dispozícií, umožňujú aplikáciu SDN vo funkčnej podobe aj v menších podnikových sietiach a preto sa očakáva ich väčší nástup.

LITERATÚRA

[1] OpenNetworking.org: *OpenFlow Switch Specification v.1.5.1* [online]. 2015, poslední aktualizace 26. 3. 2015 [cit. 27. 10. 2016]. Dostupné z URL: <https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.1.pdf>.

[2] NADEAU, Thomas D. a Kenneth GRAY. *SDN: software defined networks.* Sebastopol, CA: O'Reilly Media, 2013. ISBN 978-1-4493-4230-2.

[3] GORANSSON, Paul a Chuck BLACK. *Software defined networks: a comprehensive approach.* Amsterdam: Morgan Kaufmann, c2014. ISBN 978-0-12-416675-2

[4] ONOS: *Welcome to ONOS* [online]. 2015, poslední aktualizace 7. 12. 2015 [cit. 19. 10. 2016]. Dostupné z URL: <https://wiki.onosproject.org/pages/viewpage.action?pageId=2851517>.

[5] RYU: *RYU Controller Tutorial* [online]. 2015, poslední aktualizace 7. 7. 2015 [cit. 19. 10. 2016]. Dostupné z URL: <http://sdnhub.org/tutorials/ryu/>.

[6] TREMA: *Welcome to Trema* [online]. 2016, poslední aktualizace 26. 6. 2016 [cit. 21. 11. 2016]. Dostupné z URL: <https://github.com/trema/trema/blob/develop/README.md>.

[7] PYRETHIC: *Pyrethic* [online]. 2016, poslední aktualizace 21. 11. 2016 [cit. 21. 11. 2016]. Dostupné z URL: <http://frenetic-lang.org/pyretic/>.

[8] OPENVSWITCH: *OvS Switch* [online]. 2016, poslední aktualizace 3. 1. 2016 [cit. 3. 11. 2016]. Dostupné z URL: <http://www.openvswitch.org/>.

[9] NETSVET.CZ, KUBICA Tomáš: *OpenFlow* [online]. 2014, poslední aktualizace 17. 9. 2014 [cit. 13. 12. 2016]. Dostupné z URL: <https://www.netsvet.cz/?series=openflow>.

[10] FLOODLIGHT, IZARD Ryan: *Floodlight Documentation* [online]. 2016, poslední aktualizace 1. 5. 2016 [cit. 13. 12. 2016]. Dostupné z URL: <https://www.netsvet.cz/?series=openflow>.

[11] ingrammicroadvisor.com: BROWN Glen. *7 Advantages of Software Defined Networking* [online]. 2014, poslední aktualizace 12. 8. 2014 [cit. 13. 12. 2016]. Dostupné z URL: <http://www.ingrammicroadvisor.com/data-center/7-advantages-of-software-defined-networking>.

[12] Cariden Technologies: *Infrastrucuter SDN with Cariden Technologies* [online]. 2012, poslední aktualizace 15. 8. 2012 [cit. 13. 12. 2016]. Dostupné z URL: <http://www.sdncentral.com/wp-content/uploads/2012/08/Infrastructure_SDN.pdf>.

[13] CBENCH: *Cbench (new)* [online]. 2016, poslední aktualizace 1. 11. 2013 [cit. 20. 4. 2017]. Dostupné z URL: <https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/1343657/Cbench+New>.

[14] Google: *Software Defined Networking at Scale* [online]. 2014, poslední aktualizace 2014 [cit. 9. 5. 2017]. Dostupné z URL: <https://static.googleusercontent.com/media/research.google.com/sk//pubs/archive/42948.pdf>.

[15] techtarget.com: *Microsoft uses OpenFlow SDN for network monitoring and analysis* [online]. 2013, poslední aktualizace 17. 4. 2013 [cit. 9. 5. 2017]. Dostupné z URL: <http://searchsdn.techtarget.com/news/2240181908/Microsoft-uses-OpenFlow-SDN-for-network-monitoring-and-analysis>.

[16] techtarget.com: *The SDN implications of Facebook's open source switch* [online]. 2013, poslední aktualizace 28. 5. 2013 [cit. 9. 5. 2017]. Dostupné z URL: <http://searchsdn.techtarget.com/news/2240184752/The-SDN-implications-of-Facebooks-open-source-switch>.

ZOZNAM SYMBOLOV, VELIČÍN A SKRATIEK

SDN	Software defined networks – Softvérovo definované siete
ONF	Open Network Foundation
STP	Spanning Tree Protocol - protokol kostry grafu
RSTP	Rapid Spanning Tree Protocol - rychlý protokol kostry grafu
ICMP	Internet Control Message Protocol – protokol správy siete
IPv4	Internet protocol version 4 – Internetový protokol verzie 4
IPv6	Internet protocol version 6 – Internetový protokol verzie 6
QoS	Quality of Services – kvalita služieb
ACL	Access Control List – zoznam prístupových práv
TCAM	Ternary Content Addressable Memory – Ternárne adresovateľná pamäť
FIB	Forwarding Information Base – prepínacia tabuľka
RIB	Routing Information Base – smerovacia tabuľka
ISO	International Organization for Standardization – Medzinárodná organizácia pre normalizáciu
OSI	Open Systems Interconnection Reference Model – referenčný model OSI
API	Application programming interface – rozhranie pre programovanie aplikácií
TLS	Transport Layer Security – šifrovací protokol
TCP	Transmission Control Protocol – protokol riadenia prenosu
PPP	Point-to-Point Protocol – komunikačný protokol linkovej vrstvy
VLAN	Virtual Local Area Network – logicky nezávislá virtuálna sieť
MPLS	Multiprotocol Label Switching – typ smerovacej technológie
L2	Layer 2 – druhá vrstva OSI modelu
L3	Layer 3 – tretia vrstva OSI modelu
URI	Uniform Resource Identifier – Jednotný identifikátor zdroja
CPU	Central Processing Unit – centrálna procesorová jednotka
RAM	Random Access Memory – Pamäť s priamym prístupom

OVS	Open vSwitch – softvérový virtuálny prepínač
QOS	Quality of Services – kvalita služieb
OF	OpenFlow – protokol SDN siete

ZOZNAM PRÍLOH

A ZOZNAM SKRIPTOV MERANIA KONVERGENCIE

A.1 k-nostp-n3.py

```
#!/usr/bin/python

from mininet.net import Mininet
from mininet.node import Controller, RemoteController,
from mininet.node import OVSController, OVSSwitch
from mininet.node import Host
from mininet.cli import CLI
from mininet.log import setLogLevel, info
from mininet.link import TCLink, Intf

def emptyNet():

    net = Mininet(topo=None, build=False , ipBase='10.10.0.0/24' , link=\
        TCLink)

    info( '*** Adding controller\n' )
    net.addController( name='c0' , controller=RemoteController , ip\
        ='127.0.0.1', port=6653 )

    info( '*** Adding hosts\n' )
    h1 = net.addHost( 'h1', ip='10.10.0.1' )
    h2 = net.addHost( 'h2', ip='10.10.0.2' )
    info( '*** Adding switch\n' )
    for x in range(1, 4):
       globals()['s%s' % x] = net.addSwitch( 's%d' % (x), failMode='\
           OVSSwitch' )
    info( '*** Creating links\n' )
    net.addLink( h1, s1 )
    net.addLink( h2, s3 )
    net.addLink( s1, s3 )
    net.addLink( s2, s3 )
    net.addLink( s1, s2 )
    info( '*** Starting network\n')
    net.start()
    info( '*** Running CLI\n' )
    CLI( net )
    info( '*** Stopping network' )
    net.stop()

if __name__ == '__main__':
    setLogLevel( 'info' )
    emptyNet()
```

A.2 k-nostp-n6.py

```
#!/usr/bin/python

from mininet.net import Mininet
from mininet.node import Controller, RemoteController
from mininet.node import OVSController, OVSSwitch
from mininet.node import Host
from mininet.cli import CLI
from mininet.log import setLogLevel, info
from mininet.link import TCLink, Intf

def emptyNet():

    net = Mininet(topo=None, build=False , ipBase='10.10.0.0/24' , link=\
        TCLink)

    info( '*** Adding controller\n' )
    net.addController( name='c0' , controller=RemoteController , ip\
        ='127.0.0.1', port=6653 )

    info( '*** Adding hosts\n' )
    h1 = net.addHost( 'h1', ip='10.10.0.1' )
    h2 = net.addHost( 'h2', ip='10.10.0.2' )

    info( '*** Adding switch\n' )
    for x in range(1, 7):
       globals()['s%s' % x] = net.addSwitch( 's%d' % (x), failMode='\
           OVSSwitch' )

    info( '*** Creating links\n' )
    net.addLink( h1, s1 )
    net.addLink( h2, s6 )
    net.addLink( s1, s2 )
    net.addLink( s1, s3 )
    net.addLink( s2, s4 )
    net.addLink( s2, s5 )
    net.addLink( s3, s4 )
    net.addLink( s3, s5 )
    net.addLink( s6, s4 )
    net.addLink( s6, s5 )

    info( '*** Starting network\n')
    net.start()
    info( '*** Running CLI\n' )
    CLI( net )
```

```
42      info( '*** Stopping network' )
43      net.stop()
44
45  if __name__ == '__main__':
46      setLogLevel( 'info' )
47      emptyNet()
```

A.3 k-nostp-n10.py

```
1   #!/usr/bin/python
2
3   from mininet.net import Mininet
4   from mininet.node import Controller, RemoteController
5   from mininet.node import OVSController, OVSSwitch
6   from mininet.node import Host
7   from mininet.cli import CLI
8   from mininet.log import setLogLevel, info
9   from mininet.link import TCLink, Intf
10
11  def emptyNet():
12
13      net = Mininet(topo=None, build=False , ipBase='10.10.0.0/24' , link=\
            TCLink)
14
15      info( '*** Adding controller\n' )
16      net.addController( name='c0' , controller=RemoteController , ip\
            ='127.0.0.1', port=6653 )
17
18      info( '*** Adding hosts\n' )
19      h1 = net.addHost( 'h1', ip='10.10.0.1' )
20      h2 = net.addHost( 'h2', ip='10.10.0.2' )
21
22      info( '*** Adding switch\n' )
23      for x in range(1, 11):
24         globals()['s%s' % x] = net.addSwitch( 's%d' % (x), failMode='\
               OVSSwitch' )
25
26      info( '*** Creating links\n' )
27      net.addLink( h1, s1 )
28      net.addLink( h2, s10 )
29      net.addLink( s1, s2 )
30      net.addLink( s1, s3 )
31      for x in range(2, 4):
32         for y in range(4, 8):
33            net.addLink( 's%d' % (x), 's%d' % (y) )
34
```

```
    for x in range(8, 10):
       for y in range(4, 8):
          net.addLink( 's%d' % (x), 's%d' % (y) )
    net.addLink( s10, s8 )
    net.addLink( s10, s9 )

    info( '*** Starting network\n')
    net.start()

    info( '*** Running CLI\n' )
    CLI( net )

    info( '*** Stopping network' )
    net.stop()

if __name__ == '__main__':
    setLogLevel( 'info' )
    emptyNet()
```

A.4 k-nostp-n14.py

```
#!/usr/bin/python

from mininet.net import Mininet
from mininet.node import Controller, RemoteController
from mininet.node import OVSController, OVSSwitch
from mininet.node import Host
from mininet.cli import CLI
from mininet.log import setLogLevel, info
from mininet.link import TCLink, Intf

def emptyNet():

    net = Mininet(topo=None, build=False , ipBase='10.10.0.0/24' , link=\
        TCLink)

    info( '*** Adding controller\n' )
    net.addController( name='c0' , controller=RemoteController , ip\
        ='127.0.0.1', port=6653 )
    info( '*** Adding hosts\n' )
    h1 = net.addHost( 'h1', ip='10.10.0.1' )
    h2 = net.addHost( 'h2', ip='10.10.0.2' )
    info( '*** Adding switch\n' )
    for x in range(1, 15):
       globals()['s%s' % x] = net.addSwitch( 's%d' % (x), failMode='\
           OVSSwitch' )
```

```
23
24      info( '*** Creating links\n' )
25      net.addLink( h1, s1 )
26      net.addLink( h2, s14 )
27      net.addLink( s1, s2 )
28      net.addLink( s1, s3 )
29      for x in range(2, 4):
30         for y in range(4, 8):
31            net.addLink( 's%d' % (x), 's%d' % (y) )
32      net.addLink( s4, s8 )
33      net.addLink( s4, s9 )
34      net.addLink( s5, s8 )
35      net.addLink( s5, s9 )
36      net.addLink( s6, s10 )
37      net.addLink( s6, s11 )
38      net.addLink( s7, s10 )
39      net.addLink( s7, s11 )
40      for x in range(12, 14):
41         for y in range(8, 12):
42            net.addLink( 's%d' % (x), 's%d' % (y) )
43      net.addLink( s14, s12 )
44      net.addLink( s14, s13 )
45
46      info( '*** Starting network\n')
47      net.start()
48
49      info( '*** Running CLI\n' )
50      CLI( net )
51
52      info( '*** Stopping network' )
53      net.stop()
54
55  if __name__ == '__main__':
56      setLogLevel( 'info' )
57      emptyNet()
```

B OBSAH PRILOŽENÉHO DVD

Na priloženom DVD sa nachádzajú v jednotlivých adresároch tieto dáta:

- **DP:** Diplomová práca v elektronickej forme.
- **mininet-scripts:** Python skripty pre program Mininet, ktoré boli použité na meranie konvergencie.
- **tested-software:** Verzie kontrolérov, ktoré boli testované v diplomovej práci.
- **vm:** Virtuálny stroj stiahnutý zo stránky SDN HUB vo verzií použitej pri písaní diplomovej práce.